ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

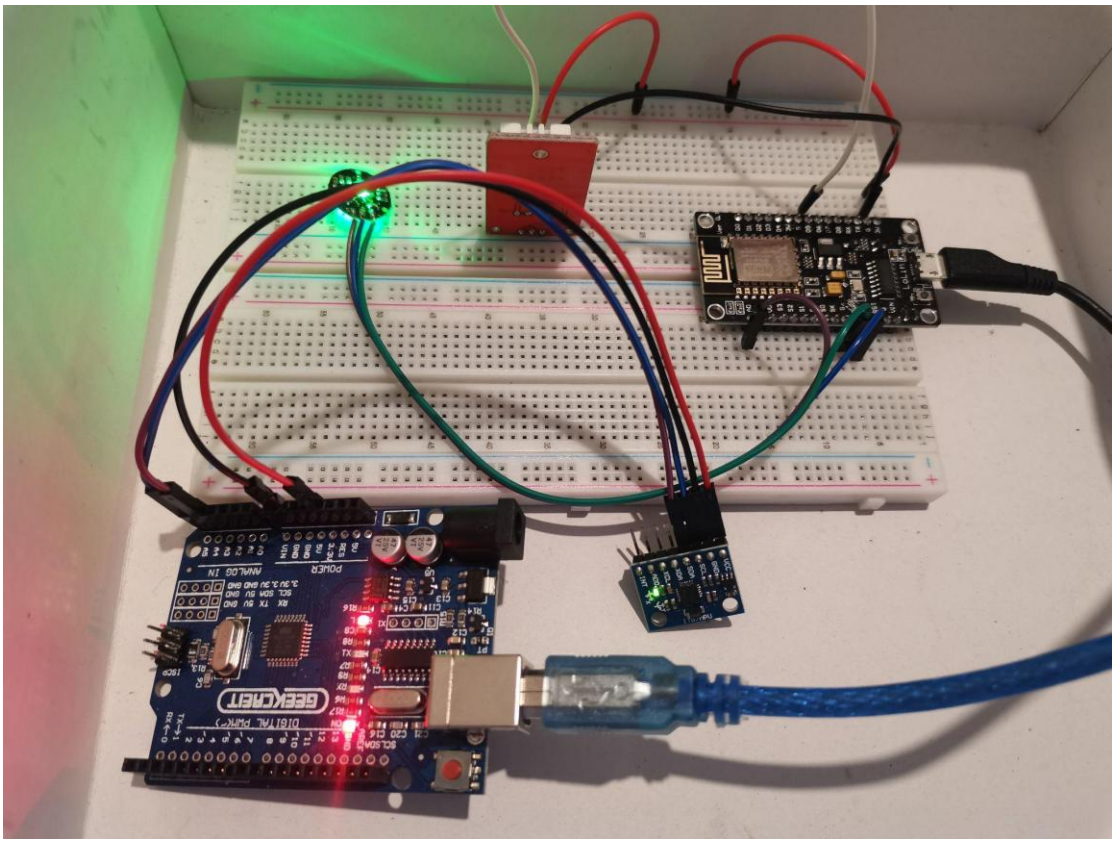
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

(ΠΡΩΗΝ ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ - ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

SmartWristband - Έξυπνο Περικάρπιο

ΚΑΤΑΜΕΤΡΗΣΗ ΘΕΡΜΟΚΡΑΣΙΑΣ ΥΓΡΑΣΙΑΣ ΚΑΙ ΒΗΜΑΤΩΝ ΜΕ ΧΡΗΣΗ ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ ARDUINO



ΣΠΟΥΔΑΣΤΗΣ : ΧΡΗΣΤΟΣ Ε. ΡΙΜΠΑΣ (Α.Μ. : 7209)

ΕΙΣΗΓΗΤΕΣ : ΛΟΥΚΑΣ ΧΑΔΕΛΛΗΣ & ΕΥΑΓΓΕΛΟΣ ΤΟΠΑΛΗΣ

ΠΑΤΡΑ, 2021

Πρόλογος

Η κεντρική ιδέα της Πτυχιακής αυτής εργασίας, αναφέρεται στο έξυπνο περικάρπιο - SmartWristband. Εκπονήθηκε στα πλαίσια του προγράμματος σπουδών του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου (πρώην Τεχνολογικό Εκπαιδευτικό Ίδρυμα Δυτικής Ελλάδας), της σχολής Μηχανικών. Σκοπός της συγκεκριμένης εργασίας είναι να παρουσιάσει τον τρόπο με τον οποίο μπορούμε να καταγράφουμε ορισμένες παραμέτρους, τόσο του οργανισμού μας, όσο και του περιβάλλοντος στο οποίο κινούμαστε. Κάτι τέτοιο είναι δυνατόν να επιτευχθεί χάρη στην πλατφόρμα Arduino προβάλλοντάς τα σε διαδικτυακή σελίδα.

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Τοπάλη Ευάγγελο και τον κ. Χαδέλλη Λουκά, καθώς και όλους τους καθηγητές οι οποίοι με βοήθησαν, ο καθένας με το δικό του τρόπο για να ολοκληρώσω την εργασία μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου καθώς και την Ευθυμία, για τη στήριξη που μου παρείχαν όλον αυτόν τον καιρό, ώστε να μπορέσω να ολοκληρώσω επιτυχώς τις σπουδές μου.

Περίληψη

Η παρούσα πτυχιακή εργασία πραγματεύεται τη μελέτη και την κατασκευή του “Smart Wristband - Έξυπνο Περικάρπιο” μέσω της υπολογιστικής πλατφόρμας ARDUINO. Στο πλαίσιο της εργασίας θα γίνουν κατανοητές έννοιες όπως η δημιουργία υλικού και λογισμικού για την ανάπτυξη ενός συστήματος που θα καταγράφει θερμοκρασία, υγρασία, καρδιακούς παλμούς και βήματα ,τα οποία θα αποστέλλονται στη διαδικτυακή σελίδα ThingSpeak (IoT) μέσω του NodeMCU όπου εκεί θα γίνονται και διάφορες αναλύσεις των δεδομένων. Ο αισθητήρας θερμοκρασίας-υγρασίας είναι ψηφιακού τύπου, ενώ αυτός των καρδιακών παλμών και των βημάτων είναι αναλογικού. Το Arduino στηρίζεται σε μικροελεγκτή της εταιρίας Atmel και αποτελεί ένα ολοκληρωμένο αναπτυξιακό σύστημα ανοιχτού λογισμικού και ευέλικτης προσαρμογής υλικού (open-source hardware), προσφέροντας δυνατότητες υλοποίησης εφαρμογών σε πολλούς τομείς της ηλεκτρονικής. Ο προγραμματισμός του γίνεται σε γλώσσα προγραμματισμού C και στο αναπτυξιακό περιβάλλον του Arduino IDE(Integrated Development Environment) το οποίο συνοδεύεται με μία βιβλιοθήκη λογισμικού που ονομάζεται “Wiring”. Το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino είναι μια εφαρμογή γραμμένη σε Java, που λειτουργεί σε πολλές πλατφόρμες και προέρχεται από το IDE για τη γλώσσα προγραμματισμού Processing και το σχέδιο Wiring. Επιπλέον, χρησιμοποιείται κι ένα πρόσθετο λογισμικό με την ονομασία Fritzing, μέσω του οποίου γίνεται αναπαράσταση των υλικών που χρησιμοποιήθηκαν για την εργασία αλλά και των αποτελεσμάτων αυτής. Το λογισμικό αυτό διαθέτει όλα τα απαραίτητα ηλεκτρονικά εξαρτήματα που χρειάζονται για τη σχεδίαση οποιουδήποτε κυκλώματος καθώς και όλες τις πλακέτες του Arduino. Για την καλύτερη κατανόηση όλων των ανωτέρω, ακολουθεί αναλυτικά η περιγραφή τους στα επόμενα κεφάλαια της εργασίας.

ABSTRACT

This dissertation deals with the study and construction of the "Smart Wristband" through the computer platform ARDUINO. The dissertation will cover concepts such as the creation of hardware and software for the development of a system that records temperature, humidity, heart rate and steps, which will be sent to the ThingSpeak (IoT) website via NodeMCU where various data analyzes will be performed. The temperature-humidity sensor is digital, while the heart rate sensor and the steps sensor are analog. The Arduino is based on an Atmel microcontroller and is an integrated open source development system and flexible open-source hardware, offering application capabilities in many areas of electronics. It is programmed in C programming language and in the development environment of the Arduino IDE (Integrated Development Environment) which is accompanied by a software library called "Wiring". The Arduino Integrated Development Environment (IDE) is a multi-platform Java-based application derived from the IDE for the Processing programming language and the Wiring design. In addition, an extra software called Fritzing is used, through which the materials used for the work and its results are represented. This software has all the necessary electronic components needed to design any circuit as well as all Arduino boards. For a better understanding of all the above, their description is detailed in the following chapters of the work.

Εισαγωγή

Σκοπός της παρούσας πτυχιακής εργασίας είναι να γίνει κατανοητός ο τρόπος λειτουργίας του περικαρπίου “ Smart Wristband ” με τη βοήθεια ενός μικροεπεξεργαστή Arduino. Για την υλοποίηση της συγκεκριμένης κατασκευής χρησιμοποιήθηκαν διάφοροι αισθητήρες, όπως καρδιακών παλμών, θερμοκρασίας-υγρασίας και βημάτων. Ωφέλιμο επίσης είναι ν’ αποτελέσει, λόγω των δυνατοτήτων του, πόλο έλξης για πολλούς οι οποίοι θα ενασχοληθούν με τη δημιουργία πολύπλοκων εφαρμογών παρέχοντας στο εν λόγω σύστημα περισσότερες τεχνικές δυνατότητες.

Η εργασία αυτή επιλέχθηκε για μια πιο πλήρη και άμεση ενημέρωση όχι μόνο των αθλούμενων αλλά και γενικότερα όλων των ατόμων που ενδιαφέρονται για τη σωματική τους κατάσταση και υγεία. Διότι μέσω του Arduino και των επιλογών του έχουν πρόσβαση σε πληροφορίες σημαντικές όπως οι καιρικές συνθήκες και οι καρδιακοί παλμοί , τα δεδομένα των οποίων θα μεταφέρονται σε ειδική ιστοσελίδα στο διαδίκτυο και θα απεικονίζονται με τη μορφή διαγραμμάτων, ενώ τα βήματα θα προβάλλονται στη σειριακή οθόνη του προγράμματος Arduino IDE. Συγκεκριμένα τα δεδομένα θα μεταφέρονται στη διαδικτυακή σελίδα ThingSpeak (IoT).

Στις μέρες μας δεν είναι λίγες οι εταιρείες οι οποίες δραστηριοποιούνται πάνω σε τέτοιες εφαρμογές. Πρόκειται λοιπόν για ένα ευρέως διαδεδομένο σύστημα το οποίο με προσέλκυσε στο να το διερευνήσω και να το κατανοήσω βαθύτερα κι έτσι πήρα την απόφαση να κατασκευάσω το “Smart Wristband - Έξυπνο Περικάρπιο”.

Περιεχόμενα

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΛΙΣΤΑ ΣΧΗΜΑΤΩΝ - ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1ο

1.1 Γενικά στοιχεία

Τα τελευταία χρόνια το διαδίκτυο κατέχει σημαντικό ρόλο στη ζωή των ανθρώπων καθώς αποτελεί το νέο μέσο ενημέρωσης και επικοινωνίας. Μέσα από αυτό η ζωή των ανθρώπων έγινε πιο εύκολη , καθώς πολλές διαδικασίες της καθημερινότητας αυτοματοποιήθηκαν και πλέον χειρίζονται απομακρυσμένα, εξοικονομώντας χρόνο. Κάτι τέτοιο είναι ιδιαίτερα ωφέλιμο καθώς οι ρυθμοί ζωής πλέον γίνονται ολοένα και γρηγορότεροι. Αυτό έχει σαν αποτέλεσμα να είναι πολύ σημαντικό η πληροφόρηση του κάθε ατόμου για μετρήσεις της καθημερινότητας, όπως για παράδειγμα οι καρδιακοί του παλμοί και τα βήματα, να γίνεται όσο το δυνατόν πιο άμεσα.

Σε ό,τι αφορά το "SmartWristband - Έξυπνο περικάρπιο" έχει σχεδιαστεί με σκοπό να παρέχει στο χρήστη πληροφορίες για τις ζωτικές παραμέτρους του οργανισμού του, όπως για παράδειγμα τους καρδιακούς του παλμούς, αλλά και στοιχεία του περιβάλλοντος όπως είναι η θερμοκρασία και η υγρασία. Τέλος, θα προβάλλει στο χρήστη τα βήματα που έχει πραγματοποιήσει.

Στα επόμενα υποκεφάλαια θα αναλυθεί λεπτομερώς το "SmartWristband - Έξυπνο περικάρπιο", τα οφέλη της γυμναστικής στην υγεία του ανθρώπου καθώς και η συχνότητα αυτής ανάλογα με την εκάστοτε ηλικία. Ακόμη, θα παρουσιαστεί ο αριθμός των καρδιακών παλμών ανάλογα με την ηλικία αλλά και με την αντίστοιχη σωματική δραστηριότητα (κατάσταση χαλάρωσης, μέση δραστηριότητα έντασης 50%-85% και έντονη σωματική δραστηριότητα).

1.2 Ορίζοντας το " SmartWristband - Έξυπνο περικάρπιο "

Πρόκειται για μία συσκευή η οποία έχει σκοπό να καταγράφει δεδομένα όπως βήματα, καρδιακούς παλμούς, θερμοκρασία - υγρασία και στη συνέχεια να τα εμφανίζει σε μία συγκεκριμένη σελίδα στο διαδίκτυο. Το "Έξυπνο περικάρπιο - SmartWristband" αναφέρεται όχι μόνο σε αθλούμενους, αλλα και σε άτομα που γενικότερα μεριμνούν για τη σωματική τους υγεία και τους παρουσιάζει με μορφή διαγραμμάτων τον καρδιακό τους ρυθμό

καθώς και τη θερμοκρασία-υγρασία. Τα βήματα του χρήστη παρουσιάζονται στη σειριακή οθόνη του Arduino IDE.

1.3 Τα οφέλη της γυμναστικής στον άνθρωπο

Από τα αρχαία κιόλας χρόνια, η γυμναστική αποτελούσε αναπόσπαστο κομμάτι της καθημερινής ρουτίνας των προγόνων μας. Η γυμναστική έχει πάρα πολλά οφέλη , τόσο για το σώμα, αλλά και για την ψυχική υγεία του ατόμου. Τα κυριότερα παρουσιάζονται παρακάτω.

1. Βελτιώνει την καρδιακή λειτουργία
2. Συμβάλλει στην ενίσχυση του ανοσοποιητικού συστήματος
3. Μειώνει το καθημερινό άγχος και στρες
4. Προστατεύει το σώμα από οστεοπόρωση
5. Αυξάνει τις επιδόσεις του οργανισμού
6. Χαρίζει ευεξία και ενέργεια
7. Μειώνει την κατάθλιψη
8. Αυξάνει την αυτοπεποίθηση
9. Ανεβάζει τα επίπεδα της καλής χοληστερίνης και συγχρόνως μειώνει τη χοληστερόλη και τα τριγλυκερίδια

Η γυμναστική που “πρέπει” να κάνει ο κάθε άνθρωπος, διαφέρει ανάλογα με τα χαρακτηριστικά του οργανισμού του, όπως για παράδειγμα την ηλικία, τη φυσική του κατάσταση και κυρίως την υγεία του. Παρακάτω, περιγράφεται πολύ σύντομα η συχνότητα της σωματικής άσκησης ανάλογα με την ηλικία.

- Ηλικίες 20-30 ετών

Είναι η ηλικία, στην οποία το σώμα βρίσκεται στο μέγιστο των δυνατοτήτων του και η συχνότητα της άσκησης που ενδείκνυται, είναι περίπου 3-4 φορές την εβδομάδα.

- Ηλικίες 30-40 ετών

Οι άνθρωποι αυτής της κατηγορίας, έχουν ανάγκη από ενδυνάμωση , καθώς ο μεταβολισμός αρχίζει σιγά-σιγά να μειώνεται. Η συχνότητα που συνιστάται, είναι 2-3 φορές την εβδομάδα.

- Ηλικίες 40-50 ετών

Σ’ αυτές τις ηλικίες, χρειάζεται σωματική άσκηση, ώστε να βοηθηθεί ο μεταβολισμός να διατηρηθεί στα φυσιολογικά του επίπεδα. Η συχνότητα για αυτές τις ηλικίες, είναι 2 φορές την εβδομάδα.

- Ηλικίες 50-60 ετών

Οι ηλικίες αυτές, χρειάζονται ασκήσεις χαμηλής έντασης 1-2 φορές την εβδομάδα.

- Ηλικίες 60 ετών και άνω

Σ’ αυτές τις ηλικίες, χρειάζεται ήπια σωματική άσκηση χωρίς πολύ κόπο. Η συχνότητα που προτείνεται λοιπόν, είναι 1-2 φορές την εβδομάδα.[1,2]

1.4 Καρδιακοί παλμοί ανά ηλικία

Οι παλμοί της καρδιάς, είναι ένας πολύ σημαντικός δείκτης υγείας που αφορά τον αριθμό των σφυγμών ανά λεπτό. Οι χαμηλοί παλμοί είναι συνήθως ένδειξη καλής υγείας, ενώ οι αυξημένοι παλμοί πολλές φορές υποδεικνύουν κάποια επιπλοκή στον οργανισμό, όπως είναι η υπόταση και σε ακραίες περιπτώσεις το εγκεφαλικό ή και το έμφραγμα. Πολλές φορές ακόμη, δημιουργούν ζαλάδα, πόνο στο στήθος και αίσθημα αδυναμίας.

Η μέτρηση των παλμών της καρδιάς είναι πολύ σημαντική και θεωρητικά μπορεί να γίνει οποιαδήποτε στιγμή της ημέρας. Ωστόσο, η μέτρηση θα είναι πιο αξιόπιστη όταν πραγματοποιηθεί αμέσως μετά το πρωινό ξύπνημα. Η “πρακτική” μέτρηση, χωρίς δηλαδή κάποιο ειδικό μηχάνημα, γίνεται τοποθετώντας το δείκτη και το μεσαίο δάχτυλο στο λαιμό, ακριβώς δίπλα από την τραχεία. Εναλλακτικά, για μέτρηση των παλμών από το χέρι, τοποθετούνται τα δύο δάχτυλα στην εσωτερική πλευρά του καρπού, ανάμεσα από το οστό και τον τένοντα, πάνω στην κερκιδική αρτηρία. Στη συνέχεια, αφού βρεθεί ο παλμός, η μέτρηση γίνεται για διάστημα 30 δευτερολέπτων και ο αριθμός που προκύπτει πολλαπλασιάζεται με το 2 για να βρεθούν οι παλμοί ανά λεπτό. Στη φωτογραφία παρακάτω, φαίνονται σχηματικά τα δύο σημεία για τη μέτρηση των παλμών.

Εικόνα 1: Τρόπος μέτρησης καρδιακών παλμών

Πηγή : Καρδιακοί Παλμοί

Οι φυσιολογικοί καρδιακοί παλμοί, διαφέρουν από άτομο σε άτομο για διάφορους παράγοντες, όπως της φυσικής κατάστασης, της ηλικίας, τα επίπεδα του στρες και της ορμονικής λειτουργίας. Για να υπολογίσει κάποιος το ανώτατο φυσιολογικό όριο των καρδιακών του παλμών, αρκεί να αφαιρέσει την ηλικία του από τον αριθμό 220. Έχει παρατηρηθεί, πως οι άνθρωποι που γυμνάζονται τακτικά, έχουν ελαφρώς μειωμένους καρδιακούς παλμούς.

Στον παρακάτω πίνακα, αναφέρονται οι φυσιολογικοί καρδιακοί παλμοί ανά ηλικία και ανά λεπτό σε επίπεδο χαλάρωσης, μέτριας(50%)-έντονης(85%) άσκησης , καθώς και το μέγιστο όριο(100%) αυτών.[3,4]

Ηλικία(έτη)	Φυσιολογικοί παλμοί σε χαλάρωση(bpm)	Φυσιολογικοί παλμοί 50-85% (bpm)	Μέγιστοι παλμοί 100% (bpm)
20	60-100	100-170	200
30	60-100	95-162	190
35	60-100	93-157	185
40	60-100	90-153	180
45	60-100	88-149	175
50	60-100	85-145	170
55	60-100	83-140	165
60	60-100	80-136	160
65	60-100	78-132	155
70	60-100	75-128	150

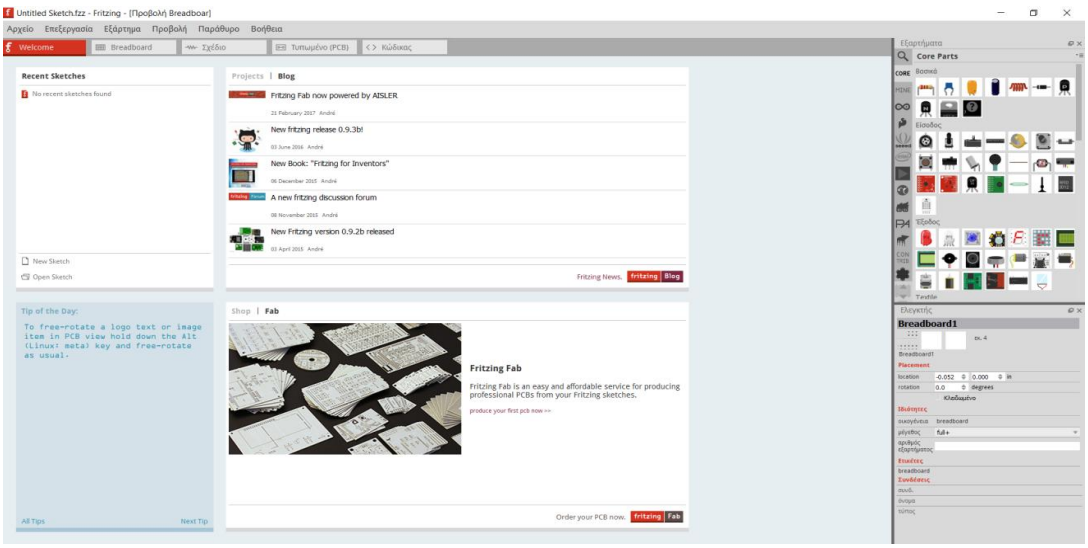
Πίνακας 1: Καρδιακοί παλμοί ανά ηλικία σε τρεις μορφές έντασης

ΚΕΦΑΛΑΙΟ 2ο

2.1 Εργαλεία ανάπτυξης λογισμικού

2.1.1 Fritzing

Όπως αναφέρθηκε παραπάνω, για την υλοποίηση της πτυχιακής εργασίας, χρησιμοποιήθηκε και το πρόγραμμα Fritzing. Πρόκειται για ένα πρόγραμμα ανοιχτού κώδικα, το οποίο επιτρέπει στο χρήστη να κατασκευάσει εικονικά το ηλεκτρονικό κύκλωμα της εφαρμογής που επιθυμεί. Το παραπάνω, αναπτύχθηκε ως μια πρωτοβουλία στο Πανεπιστήμιο Εφαρμοσμένων Επιστημών Πότσνταμ (University of Applied Sciences Potsdam). Ανεξαρτήτου κλάδου, έρευνας, χομπι ή σχεδίου, το Fritzing επιτρέπει στο χρήστη να τεκμηριώσει το πρωτότυπο έργο του που βασίζεται σε Arduino και να δημιουργήσει μια διάταξη PCB για κατασκευή. Επίσης, θεωρείται ως εργαλείο αυτοματισμού ηλεκτρονικής σχεδίασης. Από τις 2 Δεκεμβρίου του 2014, το Fritzing δίνει πρόσβαση και στον κώδικα που χρησιμοποιείται κι έτσι μπορεί ο καθένας να φορτώσει αυτόν τον κώδικα σε μια συσκευή Arduino. Βρίσκεται διαθέσιμο σε λογισμικά Windows, Unix , Mac OS X και η αρχική του σελίδα απεικονίζεται στην παρακάτω εικόνα.



Εικόνα 2: Περιβάλλον προγράμματος Fritzing

Πηγή : [Πρόγραμμα Fritzing]

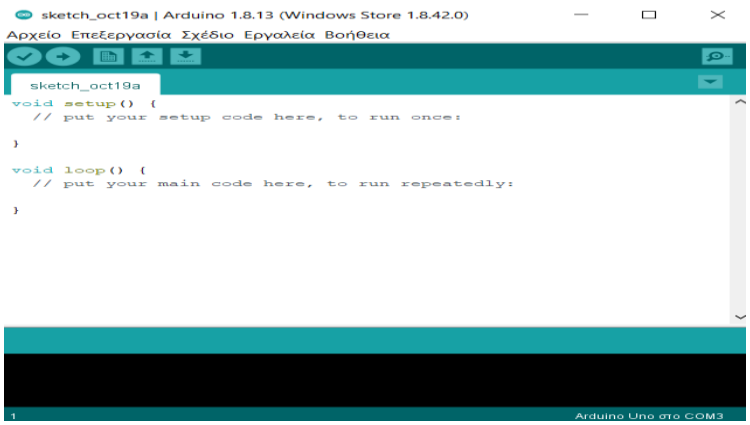
Ο πηγαίος κώδικας Fritzing, γράφεται σε C++, χρησιμοποιώντας το Qt-framework. Ο κώδικας αυτός, είναι προσβάσιμος και μπορεί τροποποιηθεί μέσω των αποθετηρίων GitHub.

Ακόμη, το FritzingFab παρέχει τη δυνατότητα στους χρήστες να παραγγέλνουν τα PCB σχέδια που αναφέρονται στο κύκλωμά τους και να εξάγει το σχέδιο σε PDF αρχείο. Ένα άλλο στοιχείο αυτού του προγράμματος που το καθιστά χρήσιμο, είναι ότι επιτρέπει στο χρήστη να δημοσιεύσει τα project του στην κεντρική σελίδα του Fritzing, με σκοπό να δοθούν εναλλακτικές ιδέες από τους υπόλοιπους χρήστες και να υπάρξει ανταλλαγή απόψεων.

Στην προηγούμενη ηλεκτρονική διεύθυνση, πέρα από το πρόγραμμα, στην επιλογή MENU, παρατηρούνται και άλλες επιλογές, όπως για παράδειγμα τα Projects των άλλων χρηστών, διάφορα ανταλλακτικά χρήσιμα για τις κατασκευές και παραδείγματα-σεμινάρια για τη σωστή χρήση του προγράμματος. Παραπάνω, φαίνεται η αρχική σελίδα του προγράμματος στην οποία, επιλέγοντας Αρχείο→ Νέο , ανοίγει μία νέα κενή σελίδα όπου ο χρήστης μπορεί να δημιουργήσει την εικονική κατασκευή του, χρησιμοποιώντας τα υλικά της επιλογής του από τη δεξιά στήλη του προγράμματος (Εξαρτήματα). Για όσα εξαρτήματα δεν υπάρχουν διαθέσιμα στις βιβλιοθήκες του προγράμματος, η προσθήκη τους γίνεται από το χρήστη κατεβάζοντας από το διαδίκτυο τις αντίστοιχες βιβλιοθήκες. Στην πάνω οριζόντια γραμμή, παρατηρούνται οι επιλογές Welcome, Breadboard, Σχέδιο, Τυπωμένο PCB και ο Κώδικας. Το πρόγραμμα, περιέχει και μερικά έτοιμα παραδείγματα για την εύκολη κατανόησή του, τα οποία βρίσκονται απο την επιλογή Αρχείο. Παρακάτω, απεικονίζονται τα 2 κυκλώματα της εργασίας.[5,6]

2.1.2 Arduino IDE

Το Arduino Uno και το NodeMCU, προκειμένου να λειτουργούν σωστά και να μπορούν να προγραμματιστούν, πρέπει να υπάρχει εγκατεστημένο στον εκάστοτε υπολογιστή το πρόγραμμα για την ανάπτυξη του κώδικα που θα επιλέξει ο χρήστης , το Arduino IDE. Το πρόγραμμα αυτό υπάρχει διαθέσιμο σε διάφορες εκδόσεις λειτουργικών συστημάτων.



Εικόνα 3: Περιβάλλον ανάπτυξης

Πηγή : Πρόγραμμα Arduino IDE

Όπως φαίνεται στην παραπάνω εικόνα, στο πλαίσιο ανάπτυξης του κώδικα υπάρχουν διαθέσιμες διάφορες επιλογές, οι οποίες θα εξηγηθούν στον πίνακα παρακάτω.[7]

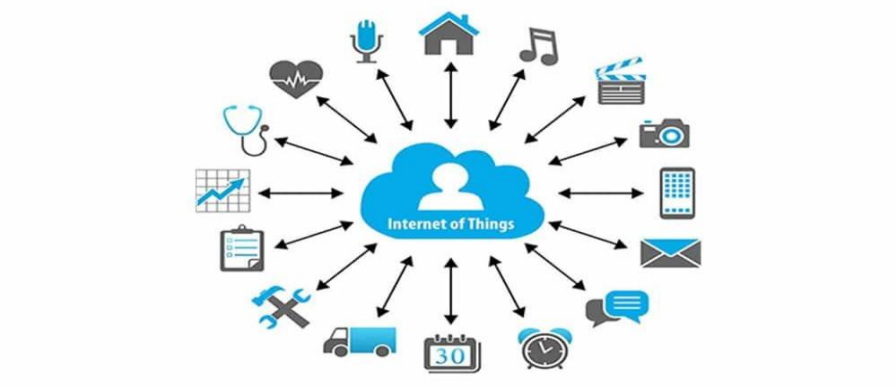
	Verify/Compile	Έλεγχος/Μεταγλώττιση για λάθη στον κώδικα
	Upload	Φορτώνει το πρόγραμμα στην πλακέτα Arduino
	New	Ανοίγει ένα νέο κενό project/sketch
	Open	Άνοιγμα αρχείου
	Save	Αποθήκευση αρχείου
	Serial Monitor	Άνοιγμα σειριακής οθόνης για προβολή δεδομένων από το πληκτρολόγιο

Πίνακας 2: Διάφορες επιλογές στο περιβάλλον ανάπτυξης κώδικα

2.1.3 Διαδίκτυο των πραγμάτων (IoT)

Το Διαδίκτυο των πραγμάτων, Internet of Things (IoT), αποτελεί το δίκτυο επικοινωνίας των συσκευών. Τέτοιες για παράδειγμα είναι οι οικιακές συσκευές, τα αυτοκίνητα, τα smartwatches, οι κάμερες καθώς και κάθε αντικείμενο το οποίο αποτελείται από ηλεκτρονικά, αισθητήρες, λογισμικό και μπορεί να έχει συνδεσιμότητα σε δίκτυο για την ανταλλαγή των δεδομένων. Αποτελεί δηλαδή ένα δίκτυο συσκευών που μεταδίδουν, διαμοιράζουν και χρησιμοποιούν δεδομένα από το φυσικό περιβάλλον, προκειμένου να παρέχουν υπηρεσίες σε πρόσωπα, εταιρείες και γενικότερα στην κοινωνία. Τα συνδεδεμένα αυτά αντικείμενα διαθέτουν μοναδικά αναγνωριστικά (Identifiers). Οι εφαρμογές του IoT είναι πάρα πολλές και μπορούν να αναφέρονται σε διάφορους κλάδους όπως της υγείας, των μεταφορών και του περιβάλλοντος. Βασικό χαρακτηριστικό όλων αυτών των συσκευών είναι η σύνδεσή τους με σκοπό να παρέχουν δυνατότητα στο χρήστη να τα ελέγχει από έναν υπολογιστή ή κινητό. Ο όρος Internet of Things δόθηκε τη δεκαετία του 1990 απο τον Kevin

Ashton, ο οποίος ήταν αυτός που ανακάλυψε τον τρόπο να συνδέσει αντικείμενα με το διαδίκτυο μέσω μιας ετικέτας RFID και ήταν ένας από τους ιδρυτές του Auto-ID center στο MIT.

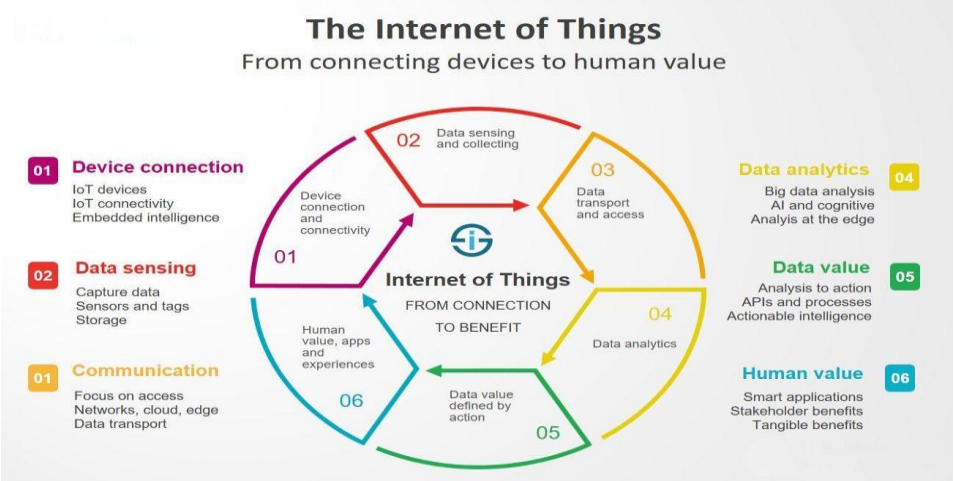


Εικόνα 4: Πληθώρα συσκευών και Internet of Things

Πηγή : Internet of Things και συσκευές

Ο τρόπος με τον οποίο λειτουργεί το Διαδίκτυο των πραγμάτων περιγράφεται παρακάτω. Οι συσκευές και τα αντικείμενα με ενσωματωμένους αισθητήρες συνδέονται σε μια πλατφόρμα, η οποία περιλαμβάνει δεδομένα από τις διάφορες συσκευές και με την ανάλυσή τους μοιράζονται τις πιο πολύτιμες πληροφορίες με εφαρμογές που έχουν δημιουργηθεί για την αντιμετώπιση συγκεκριμένων αναγκών. Οι συσκευές IoT μπορούν να εντοπίσουν με μεγάλη ακρίβεια ποιές πληροφορίες είναι χρήσιμες και στη συνέχεια να τις εκμεταλλευτούν κατάλληλα. Με τη διαδικασία αυτή εξοικονομείται χρόνος, αφού πολλές διεργασίες είναι επαναλαμβανόμενες.

Η χρησιμότητα αυτής της IoT πλατφόρμας είναι τεράστια, αφού ολοένα αυξάνει και η ζήτησή της από τους αγοραστές. Αυτό γίνεται γιατί οι άνθρωποι προσπαθούν να επιτύχουν την αυτονομία σε πολλά πράγματα μέσα στην καθημερινότητά τους. Μερικά παραδείγματα είναι η ενεργοποίηση του κλιματιστικού χωρίς να παρευρίσκεται ο χρήστης στο χώρο ,ένα έξυπνο ψυγείο που θα τον ενημερώνει για βασικές ελλείψεις προϊόντων και σε μεγαλύτερης κλίμακας παράδειγμα, έχουν προχωρήσει οι Η.Π.Α. με την τοποθέτηση έξυπνων μετρητών νερού, κάτι το οποίο είχε ως αποτέλεσμα την άμεση και ασφαλή εξοικονόμηση χρημάτων.[8]

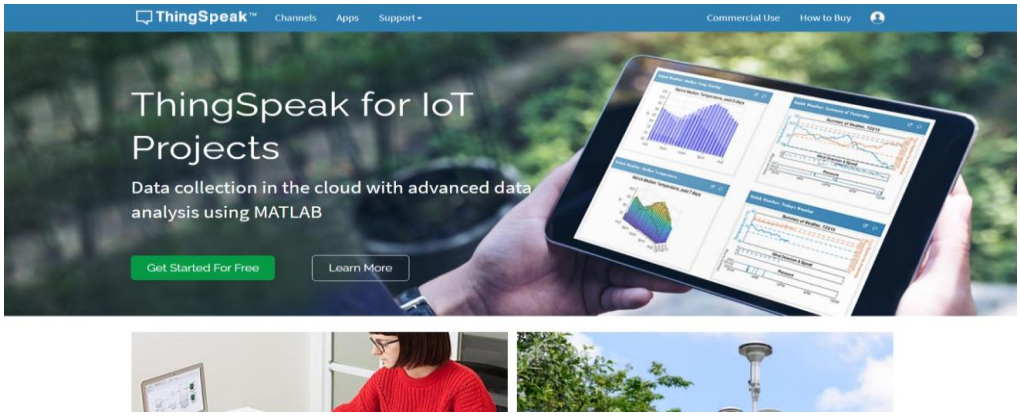


Εικόνα 5: Τρόπος λειτουργίας του Internet of Things

Πηγή : Τρόπος λειτουργίας του Internet of Things

2.1.4 ThingSpeak

Για τη συλλογή των δεδομένων των διαφόρων αισθητήρων της εργασίας, χρησιμοποιήθηκε η πλατφόρμα “ThingSpeak Internet of Things - IoT Analytics”. Σύμφωνα με τους προγραμματιστές της πλατφόρμας, αποτελεί μία εφαρμογή ανοιχτού κώδικα και API (Application Programming Interface) για την αποθήκευση και την ανάκτηση των δεδομένων, χρησιμοποιώντας το πρωτόκολλο HTTP και MQTT, μέσω του διαδικτύου ή ενός τοπικού δικτύου. Το ThingSpeak κυκλοφόρησε αρχικά από το ioBridge το 2010 , ως υπηρεσία υποστήριξης εφαρμογών IoT (Internet of Things) . Η γλώσσα που χρησιμοποιήθηκε για την συγγραφή της είναι η Ruby, η οποία αποτελεί γλώσσα προγραμματισμού υψηλού επιπέδου γενικού σκοπού. Σχεδιάστηκε και αναπτύχθηκε στα μέσα της δεκαετίας του 1990 απο τον Yukihiro “Matz” Matsumoto στην Ιαπωνία. Διαθέτει ενσωματωμένη υποστήριξη από το λογισμικό αριθμητικών υπολογιστών MATLAB, της MathWorks. Έτσι επιτρέπει στους χρήστες της πλατφόρμας να αναλύουν και να οπτικοποιούν τα μεταφορτωμένα δεδομένα, χρησιμοποιώντας ουσιαστικά το MATLAB χωρίς να απαιτείται η αγορά άδειας από τη MathWorks. Το ThingSpeak, έχει άμεση σχέση με τη MathWorks, για το λόγο ότι ένας χρήστης έχοντας λογαριασμό στη MathWorks, μπορεί να είναι χρήστης και του ThingSpeak με τον ίδιο λογαριασμό.[9]

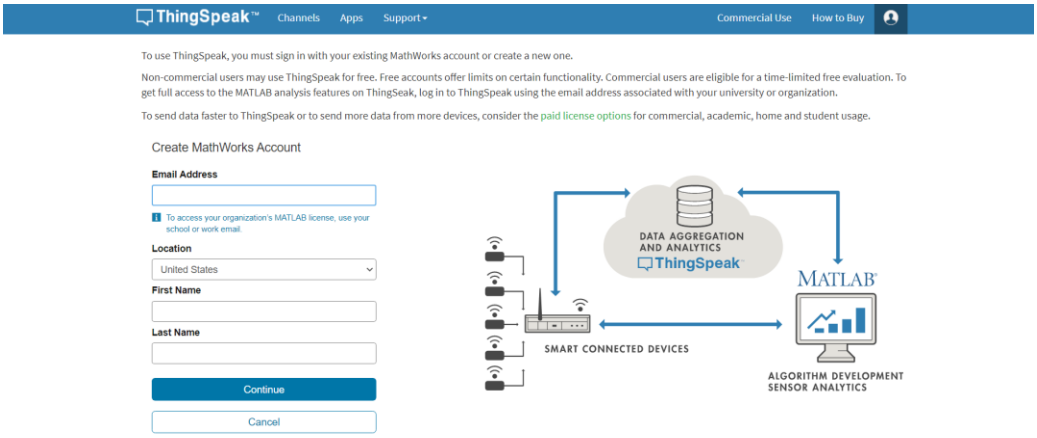


Εικόνα 6: Αρχική σελίδα προγράμματος ThingSpeak

Πηγή : [Αρχική σελίδα ThingSpeak IoT]

Για τη χρήση του προγράμματος και των λειτουργιών που παρέχει πρέπει να γίνει η εγγραφή του χρήστη και η προσθήκη των αισθητήρων της εκάστοτε εφαρμογής, προκειμένου να γίνεται η επεξεργασία των δεδομένων και η απεικόνισή τους. Αυτή η διαδικασία, περιγράφεται παρακάτω.

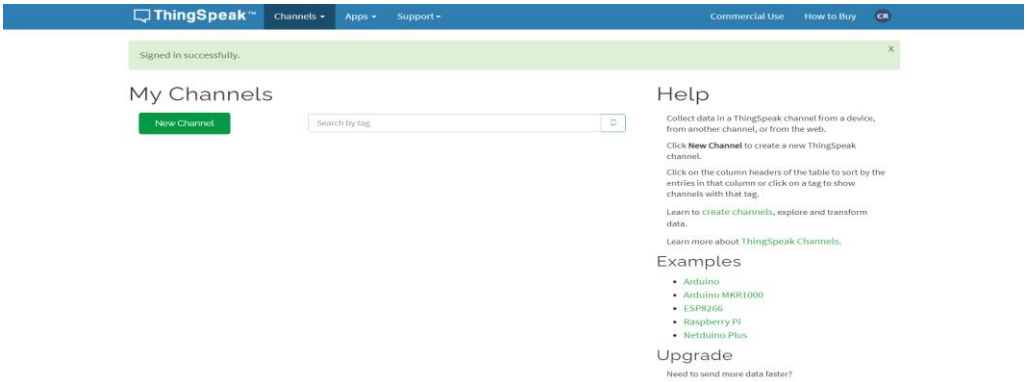
Από την αρχική σελίδα του προγράμματος, επιλέγουμε το Sign In στο πάνω δεξιό μέρος της καρτέλας και στη συνέχεια το Create One για τη δημιουργία του λογαριασμού.[10]



Εικόνα 7: Διαδικασία δημιουργίας λογαριασμού στο ThingSpeak

Πηγή : [Δημιουργία λογαριασμού στο ThingSpeak]

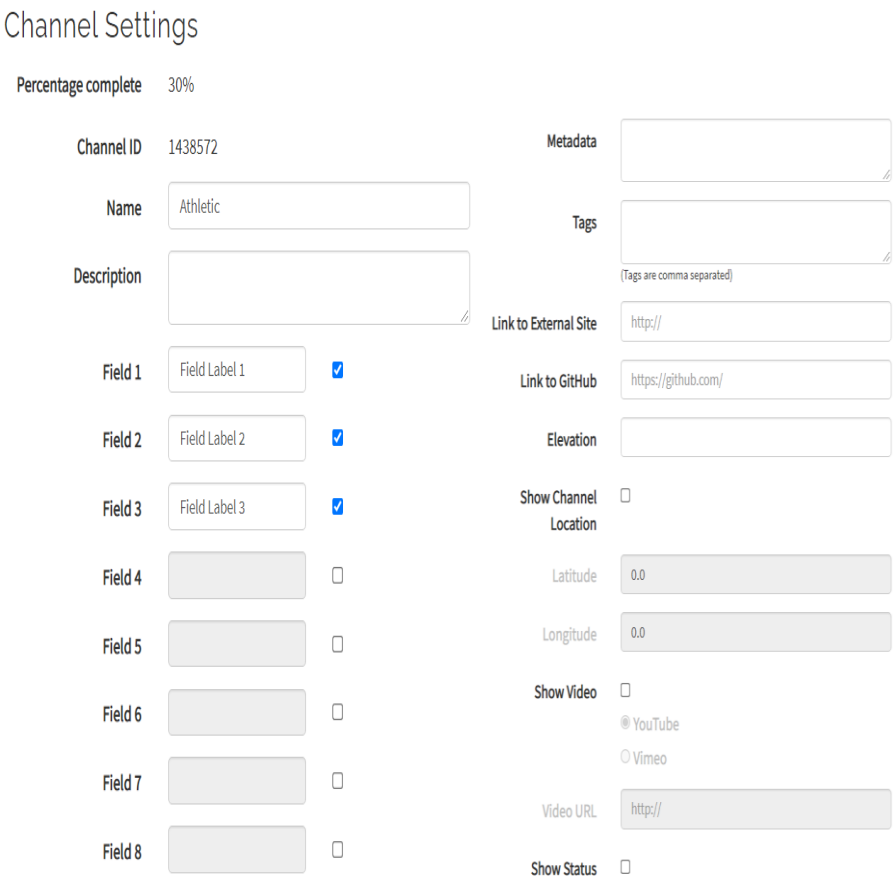
Έχοντας συμπληρώσει σωστά τα στοιχεία μας στα παραπάνω πεδία, μπορούμε να αρχίσουμε να προσθέτουμε τους αισθητήρες της εφαρμογής μας , επιλέγοντας New Channel στην αριστερή μεριά της σελίδας.



Εικόνα 8: Διαδικασία προσθήκης αισθητήρων

Πηγή : [Προσθήκη αισθητήρων σε ThingSpeak IoT]

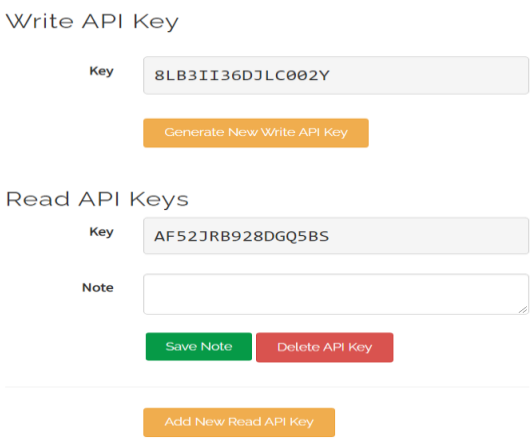
Στο επόμενο παράθυρο της σελίδας , θα αρχίσουμε να προσθέτουμε τους αισθητήρες της εφαρμογής μας, τον τύπο των δεδομένων τους καθώς και τη ρύθμιση για την ανάλυση των δεδομένων που επιθυμούμε. Στην παρακάτω φωτογραφία απεικονίζονται οι ρυθμίσεις που έχουν τεθεί για τους αισθητήρες θερμοκρασίας - υγρασίας και καρδιακών παλμών.[11]



Εικόνα 9: Αναλυτικά χαρακτηριστικά των αισθητήρων

Πηγή : [Χαρακτηριστικά αισθητήρων]

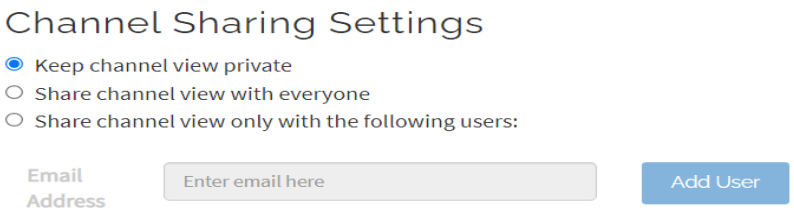
Το κανάλι δεδομένων του ThingSpeak συλλέγει τα δεδομένα από την εκάστοτε εφαρμογή. Κάθε κανάλι περιλαμβάνει 8 πεδία - Fields τα οποία μπορούν να αποθηκεύουν δεδομένα για 8 διαφορετικούς τύπους. Ακόμη, περιέχει 3 πεδία - Fields για δεδομένα που αφορούν τοποθεσία και 1 πεδίο - Field που αφορά δεδομένα κατάστασης. Έχοντας συλλέξει τα δεδομένα σε ένα κανάλι, μπορούμε στη συνέχεια να χρησιμοποιήσουμε τις επιλογές που μας προσφέρει το ThingSpeak για την ανάλυση και την απεικόνισή τους. Οι ρυθμίσεις για το κάθε κανάλι αφορούν την ονομασία του, την περιγραφή του, τον τύπο δεδομένων που θα συλλέγονται στο κανάλι και την τοποθεσία του αισθητήρα με χαρακτηριστικά το υψόμετρο, το γεωγραφικό πλάτος και μήκος της περιοχής. Επιπλέον μπορεί να προστεθεί κάποιο Link, το οποίο θα μεταφέρει το χρήστη σε κάποιο δεύτερο Site, δίνοντάς του περαιτέρω πληροφορίες για το κανάλι των δεδομένων. Ρυθμίσεις επίσης μπορούν να πραγματοποιηθούν και στα Charts, στα διαγράμματα δηλαδή που θα εμφανίζονται οι τιμές των αισθητήρων. Μετά τη ρύθμιση των παραπάνω πεδίων, επιλέγουμε Save Channel για να αποθηκεύσουμε το κανάλι με τις ρυθμίσεις μας. Στην καρτέλα API keys της σελίδας του ThingSpeak θα συναντήσουμε το “Write API Key”, το οποίο χρησιμοποιείται για την εγγραφή δεδομένων σε ένα κανάλι και το “Read API Keys” , το οποίο χρησιμοποιείται για την προβολή δεδομένων ή διαγραμμάτων σε έναν άλλο χρήστη.



Εικόνα 10 : Κλειδιά ThingSpeak

Πηγή : Κλειδιά εγγραφής-ανάγνωσης δεδομένων

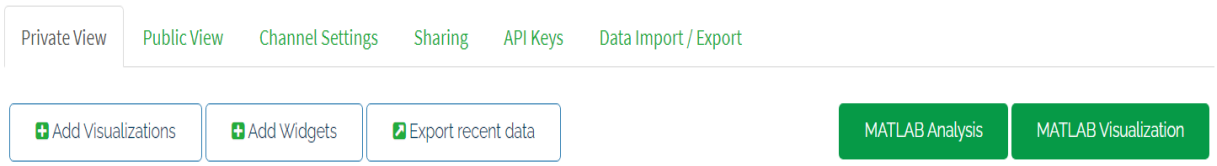
Στην καρτέλα Sharing της σελίδας του ThingSpeak, υπάρχει η δυνατότητα επιλογής σε ποιον θα προβληθούν τα δεδομένα του καναλιού. Στη δική μας περίπτωση έχουμε επιλέξει την ιδιωτική προβολή των δεδομένων.



Εικόνα 11 : Επιλογές ιδιωτικότητας του καναλιού

Πηγή : Ιδιωτικότητα καναλιού ThingSpeak

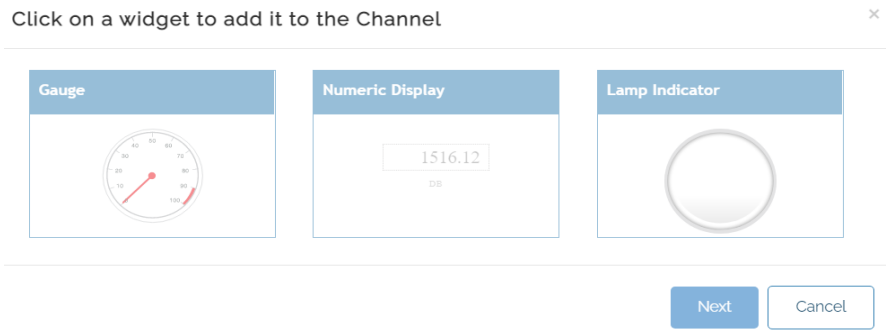
Στην καρτέλα "Private View" , υπάρχουν διαθέσιμες οι επιλογές " Add Visualizations" , "Add Widgets" και η επιλογή " Export recent data " όπως φαίνονται παρακάτω. Στη δική μας περίπτωση, προσθέσαμε μερικά Widgets για τη θερμοκρασία, την υγρασία και τους καρδιακούς παλμούς.



Εικόνα 12 : Επιλογές διαχείρισης των δεδομένων του καναλιού

Πηγή : Διαχείριση δεδομένων καναλιού

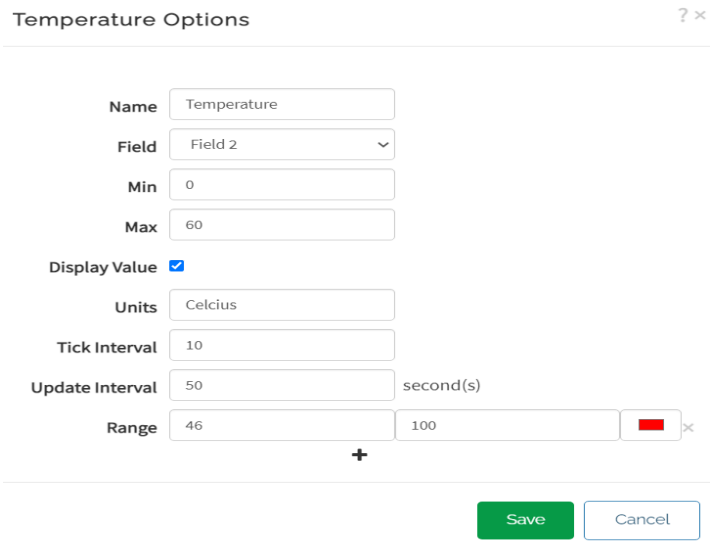
Επιλέγοντας το "Add Widgets" μας εμφανίζει την παρακάτω εικόνα με τις διαθέσιμες επιλογές. Στη δική μας περίπτωση, προσθέσαμε ένα widget τύπου Gauge για την ένδειξη της θερμοκρασίας, της υγρασίας και των καρδιακών παλμών.



Εικόνα 13 : Widget Gauge, Numeric Display, Lamp Indicator

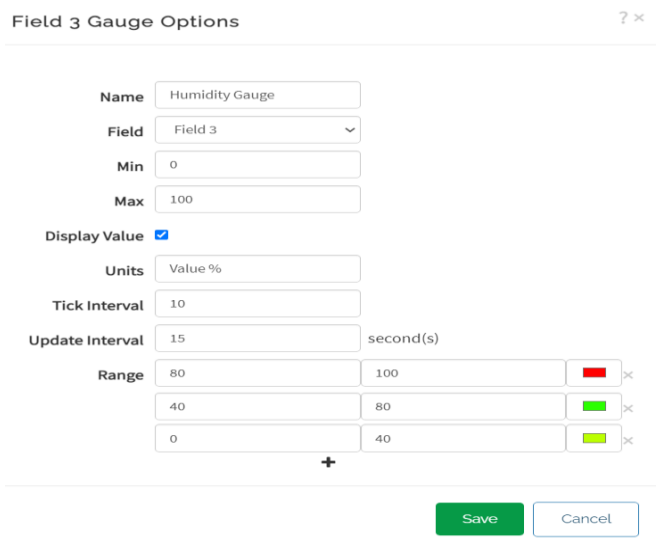
Πηγή : Gauge Widget, Numeric Display, Lamp Indicator

Στη συνέχεια, επιλέγοντας στις ρυθμίσεις του εκάστοτε Widget θα μας εμφανίσει το αντίστοιχο παράθυρο με τις ρυθμίσεις για την ομαλή λειτουργία του. Επιλέγοντας “Save” οι ρυθμίσεις έχουν αποθηκευτεί στο κανάλι.



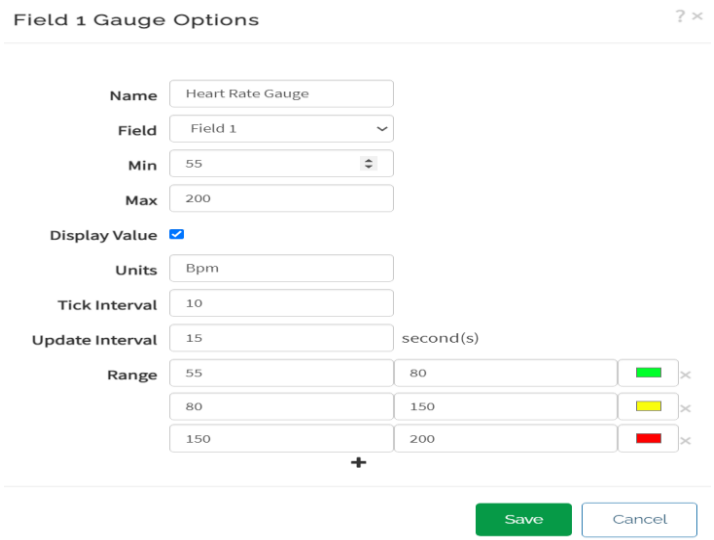
Εικόνα 14 : Ρυθμίσεις Widget θερμοκρασίας

Πηγή : Ρυθμίσεις Widget θερμοκρασίας



Εικόνα 15 : Ρυθμίσεις Widget υγρασίας

Πηγή : Ρυθμίσεις Widget υγρασίας



Εικόνα 16 : Ρυθμίσεις Widget καρδιακών παλμών

Πηγή : Ρυθμίσεις Widget καρδιακών παλμών

2.1.5 Επικοινωνία NodeMCU με ThingSpeak (IoT)

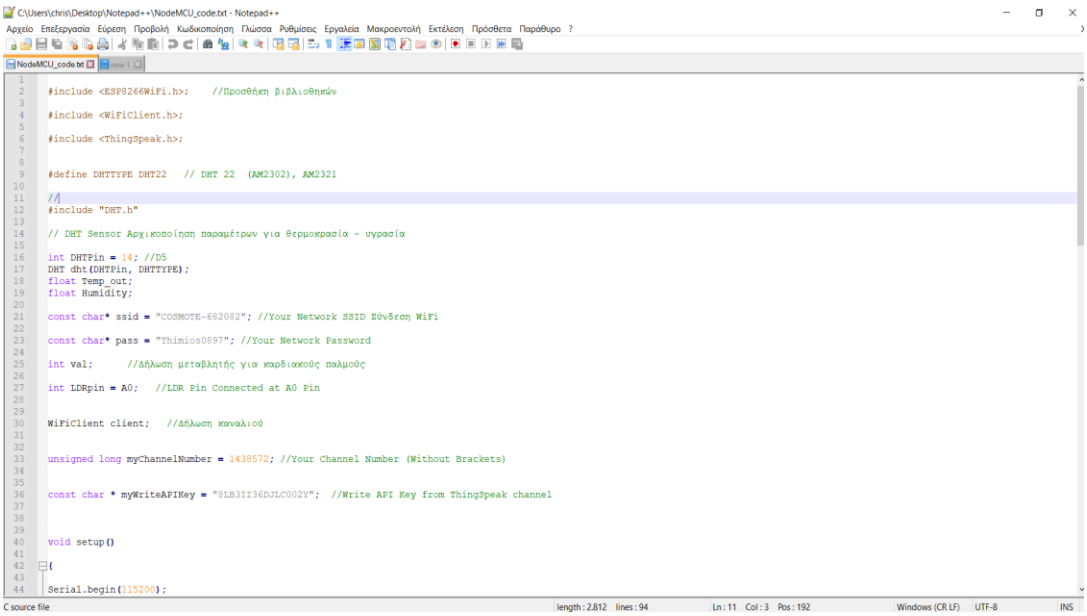
Για τη μεταφορά των δεδομένων των αισθητήρων στην πλατφόρμα ThingSpeak (IoT) χρησιμοποιήθηκε μία αυτόνομη μονάδα μικροελεγκτή με την ονομασία NodeMCU(Micro-Controller Unit). Με τον όρο αυτόνομη μονάδα μικροελεγκτή εννοούμε έναν μικροελεγκτή ο οποίος είναι προσαρμοσμένος σε μία πλακέτα τυπωμένου κυκλώματος(PCB). Η πλακέτα αυτή περιλαμβάνει όλα τα απαραίτητα κυκλώματα για μία εργασία ελέγχου. Τέτοια είναι ο μικροεπεξεργαστής, τα κυκλώματα εισόδου - εξόδου (I/O), η γεννήτρια ρολογιού, η μνήμη τυχαίας προσπέλασης (RAM), το αποθηκευμένο σε μνήμη πρόγραμμα και τυχόν απαραίτητα υποστηρικτικά IC. Μια τέτοια πλακέτα είναι άμεσα διαθέσιμη για χρήση από έναν προγραμματιστή, χωρίς να χρειάζεται να αφιερώσει χρόνο για τη δημιουργία του υλικού του ελεγκτή. Δεδομένου του χαμηλού κόστους που διαθέτουν, είναι ευρέως γνωστοί στο χώρο της εκπαίδευσης και αποτελούν ένα δημοφιλές μέσο για τους προγραμματιστές ώστε να αποκτήσουν πρακτική εμπειρία με μία νέα οικογένεια επεξεργαστών.

Το NodeMCU είναι μία IoT πλατφόρμα ανοιχτού κώδικα με χαμηλό κόστος. Περιλαμβάνει υλικολογισμικό το οποίο “τρέχει” με το ESP8266 Wi-Fi Soc(System On a Chip) από την Espressif Systems και υλικό-hardware το οποίο βασίζεται στη μονάδα ESP-12. Η ονομασία NodeMCU, συνδυάζει τον Κόμβο - Node και τη Μονάδα Μικροελεγκτή - Micro-Controller Unit. Το υλικολογισμικό και τα πρότυπα σχέδια πλακέτας είναι ανοιχτά για τον κάθε χρήστη, σε περίπτωση που επιθυμεί να προβεί σε περαιτέρω ενέργειες και ρυθμίσεις

ανάλογα με την εφαρμογή του. Το υλικολογισμικό - firmware χρησιμοποιεί τη γλώσσα προγραμματισμού Lua και είναι βασισμένο στο Espressif Non-OS SDK. Λόγω των περιορισμένων πόρων, οι χρήστες πρέπει να επιλέξουν τις ενότητες που σχετίζονται με το έργο τους και να δημιουργήσουν ένα υλικολογισμικό προσαρμοσμένο στις ανάγκες τους. Ο σχεδιασμός βασίστηκε κυρίως στη μονάδα ESP-12 του ESP8266, η οποία έχει ένα Wi-Fi SoC ενσωματωμένο με έναν πυρήνα Tensilica Xtensa LX106, ο οποίος χρησιμοποιείται σε πολλές IoT εφαρμογές.[12,13,14]

2.1.6 Notepad ++

Το Notepad ++ πρόκειται για το πιο γνωστό πρόγραμμα ανοιχτού λογισμικού για τη συγγραφή και επεξεργασία κειμένου και κώδικα. Χρησιμοποιεί το στοιχείο επεξεργασίας Scintilla και έχει κερδίσει δύο φορές το βραβείο SourceForge Community Choice για το καλύτερο εργαλείο προγραμματιστή. Η πρώτη του έκδοση δημοσιεύτηκε το 2003, από τον Don Ho και από το 2015 φιλοξενείται στο GitHub. Είναι εφαρμογή μόνο για Microsoft Windows καθώς απορρίφθηκε η πρόταση για επέκταση της εφαρμογής και στις πλατφόρμες Mac Os X και Unix. Πρόκειται για ένα πολύ εύχρηστο πρόγραμμα με πληθώρα επιλογών προς το χρήστη. Στο επάνω μέρος της εφαρμογής υπάρχει μία εργαλειοθήκη με τις κυριότερες επιλογές , όπως για παράδειγμα τις ρυθμίσεις για την προβολή του κώδικα, τη γλώσσα που είναι γραμμένος ο κώδικας καθώς και διάφορα άλλα εργαλεία. Το μεγαλύτερο κομμάτι της εφαρμογής καλύπτεται από το χώρο για τη συγγραφή του κώδικα. Το βασικό χαρακτηριστικό αυτής της εφαρμογής είναι η επιλογή syntax highlighting , η οποία χρωματίζει τον κώδικα ανάλογα με τη δομή του και τη γλώσσα προγραμματισμού που έχει επιλεγεί.[16,17]

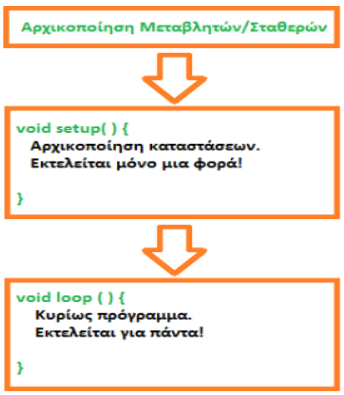


Εικόνα 17 : Περιβάλλον του Notepad++

Πηγή : Πρόγραμμα Notepad++

2.2 Βασικές εντολές

Το πρόγραμμα χωρίζεται σε τρία μέρη : τη δομή, τις μεταβλητές/σταθερές και τις συναρτήσεις. Η δομή με τη σειρά της , χωρίζεται κι αυτή σε τρία μερη : τη δήλωση των μεταβλητών, το κομμάτι του κώδικα που περιέχει την αρχικοποίηση καταστάσεων και μεταβλητών , καθώς και τον κώδικα που θέλουμε να τρέξει μία μόνο φορά στο Arduino και το κομμάτι του κώδικα loop() , το οποίο περιέχει το κυρίως πρόγραμμα μας και θα τρέχει συνέχεια, μέχρι να αποσυνδέσουμε το Arduino από την πηγή τροφοδοσίας. Το σχήμα παρακάτω αναφέρεται στο κομμάτι της δομής.



Εικόνα 18 : Κύρια μέρη προγράμματος

Πηγή : Κύρια μέρη προγράμματος

Πιο αναλυτικά με την παραπάνω εικόνα, παρατηρούνται δύο ειδικές συναρτήσεις που είναι μέρος του κάθε sketch του Arduino, οι οποίες είναι η **setup()** και η **loop().** Η εντολή setup(), καλείται μόνο μία φορα ή όταν κάνει reset η πλατφόρμα Arduino. Σε αυτή γίνονται οι αρχικοποιήσεις των μεταβλητών , η ρύθμιση της κατάστασης των ακίδων (pins) και η προετοιμασία των βιβλιοθηκών. Σε αντίθεση με τη setup() βρίσκεται η εντολή loop(), η οποία καλείται ξανά και ξανά, επιτρέποντας έτσι στο πρόγραμμα να ανταποκριθεί σε εξωτερικά ερεθίσματα. Οι δύο αυτές συναρτήσεις πρέπει να περιλαμβάνονται στο sketch ακόμη και αν δεν περιέχουν κάτι και είναι κενές.

Οι μεταβλητές με τη σειρά τους παίζουν πολύ σημαντικό ρόλο στη σύνταξη του κώδικα, καθώς αλλάζοντας τις τιμές τους κατά την εκτέλεση του προγράμματος, μπορούν να επιτευχθούν διάφορες λειτουργίες. Μπορούν να δεχθούν διάφορες τιμές, όπως χαρακτήρες,νούμερα, χαρακτήρες και νούμερα ή να έχουν κάποια λογική τιμή, όπως για παράδειγμα True / False (Αληθής/ Ψευδής). Ανάλογα με την τιμή αυτή, γίνεται η αρχικοποίησή τους ή η δήλωσή τους αντίστοιχα στο πρώτο τμήμα της δομής του προγράμματος , με τον εξής τρόπο : τύπος όνομα_μεταβλητής.[18]

int number: Ακέραιους αριθμούς από το -32768 μέχρι το 32768
π.χ. int number= 500; ή int arnitikos= - 5;
float number1: Δεκαδικούς αριθμούς
π.χ. float number1= 0.000014; ή float number_2= 12.3476
char onoma γράμματα αλλά και αριθμούς
π.χ. char onoma=’Christos’; ή char gramma=’C’; ή char_asxeto=’ACH8t’;
boolean state Τιμή true ή false
π.χ. boolean state1= true; ή boolean state2= false;

Κάθε όνομα που δίνεται σε μια μεταβλητή ή σταθερή πρέπει να υπάρχει μόνο μία φορά μέσα στον κώδικα. Το όνομα πρέπει να είναι με αγγλικούς χαρακτήρες και να μην ξεκινάει με νούμερο.Όπως φαίνεται και από τα παραπάνω παραδείγματα, μετά την κάθε μεταβλητή χρησιμοποιείται το ελληνικό ερωτηματικό ‘;’ σα χαρακτήρας τερματισμού.Στις σταθερές ισχύει το ίδιο, αλλά πριν τον τύπο γράφεται ο χαρακτηρισμός ‘const’ (const τύπος όνομα_σταθερής). Μια σταθερή χρησιμοποιείται σα συντόμευση μέσα στο πρόγραμμα, αποφεύγοντας έτσι τιμές τις οποίες μπορεί να ξεχαστούν παρακάτω, όπως για παράδειγμα const int statheri_timi=97 ; . Μια σταθερή τιμή υποχρεωτικά πρέπει να έχει σταθερή τιμή, σε αντίθεση με άλλες μεταβλητές, των οποίων η τιμή μπορεί να προκύπτει με την εκτέλεση

κάποιας εντολής μέσα στο πρόγραμμα. Αυτή η ποικιλία των συναρτήσεων είναι υπεύθυνη για την επικοινωνία του Arduino Uno με τα υπόλοιπα στοιχεία του κυκλώματος, όπως τα led , τα κουμπιά, οι διακόπτες, η οθόνη LCD. Οι βασικότερες από αυτές, παρουσιάζονται παρακάτω.

- **pinMode(pin,mode);**

Αποτελεί πολύ σημαντικό παράγοντα για το Arduino να δηλωθεί εάν ο ακροδέκτης που έχει συνδεθεί με το στοιχείο του κυκλώματός , είναι είσοδος ή έξοδος, δηλαδή αν θα λάβει ή αν θα στείλει κάποιο σήμα. Για παράδειγμα αν έχει συνδεθεί κάποιο led, τότε στέλνεται σήμα για να ανάψει ή να σβήσει, άρα το pin που είναι συνδεδεμένο στο Arduino Uno , πρέπει να δηλωθεί ως έξοδο. Σε αντίθετη περίπτωση αν είναι συνδεδεμένο ένα κουμπί-διακόπτης, θα ληφθεί σήμα στο Arduino uno όταν ενεργοποιηθεί για να εκτελεστεί κάποια λειτουργία, άρα το pin θα δηλωθεί ως είσοδος, π.χ. pinMode(5,INPUT); ή pinMode(2,OUTPUT);

- **digitalWrite(pin, value) ;**

Αφού έχει δηλωθεί ο ρόλος του pin που χρειάζεται με την εντολή pinMode, πρέπει να δηλωθεί στο Arduino τί να εκτελέσει με αυτό. Αν έχει τοποθετηθεί στο Arduino ένα led με σκοπό να ανάψει, θα γίνει χρήση της συνάρτησης digitalWrite(pin_τοποθέτησης,HIGH); . Για να σβήσει το led, τότε χρησιμοποιείται η συνάρτηση digitalWrite(pin_τοποθέτησης,LOW); . Με την τιμή HIGH, δίνουμε λογικό ‘1’, συνεπώς τάση 5 Volt-Vcc και με την τιμή LOW, δίνουμε λογικό ‘0’ , συνεπώς γείωση-Gnd.

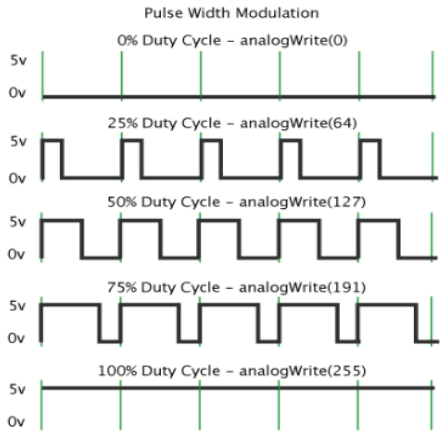
- **όνομα_μεταβλητής=digitalRead(pin);**

Με την εντολή digitalRead ουσιαστικά το Arduino διαβάζει μια τιμή. Η τιμή αυτή μπορεί επίσης να είναι HIGH ή LOW. Σε ένα κουμπί για παράδειγμα που περνάει ρεύμα το Arduino λαμβάνει την εξής τιμή : timi=digitalRead(pin_τοποθέτησης); με την τιμή να είναι HIGH. Ενώ, όταν το κουμπί πατηθεί, η τιμή θα γίνει LOW.

- **analogWrite(pin,value);**

Με την προηγούμενη εντολή δίνονται μόνο δύο τιμές HIGH , LOW. Ωστόσο, υπάρχουν εφαρμογές όπως ο χαμηλός φωτισμός σε ένα led ή η μείωση της ταχύτητας περιστροφής ενός DC μοτέρ, οι οποίες απαιτούν την ύπαρξη συνάρτησης που να μπορεί να δεχθεί άλλες τιμές. Με τη συνάρτηση analogWrite μπορούν να δοθούν τιμές από 0 έως 255. Η λειτουργία αυτή ονομάζεται Pulse Width Modulation- PWM και στο Arduino μπορεί να χρησιμοποιηθεί στα pin 3,5,6,9,10 και 11. Ουσιαστικά αυτή η λειτουργία στέλνει παλμούς. Μοιάζει δηλαδή με

έναν διακόπτη , ο οποίος ανοιγοκλείνει πολύ γρήγορα. Για παράδειγμα στην εφαρμογή μείωσης ταχύτητας ενός DC μοτέρ στη μισή, χρησιμοποιείται η εντολή : analogWrite(pinPWM_τοποθέτησης,127); . Η παρακάτω εικόνα παρουσιάζει σχηματικά ότι αναφέρθηκε προηγουμένως χρησιμοποιώντας 5 τιμές για την περιοχή τιμών από 0-255.



Εικόνα 19: PWM Modulation-Διαμόρφωση εύρους παλμών

Πηγή : Διαμόρφωση εύρους παλμών - PWM

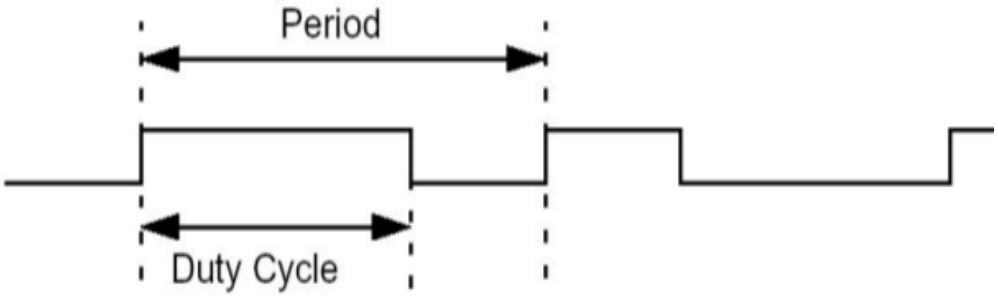
- **όνομα_μεταβλητής=analogRead(pin);**

Με τη συγκεκριμένη συνάρτηση το Arduino μπορεί να λάβει μία αναλογική τιμή, όπως για παράδειγμα από έναν αισθητήρα θερμοκρασίας ή αισθητήρα υπερύθρων. Η περιοχή τιμών σε αυτή την περίπτωση είναι από 0 έως 1023. Τα διάφορα αναλογικά εξαρτήματα που θα συνδεθούν, πρέπει να τοποθετηθούν στις θέσεις A0-A5.

- **delay()=ms;**

Με την εντολή αυτή το πρόγραμμα παύει για όσο χρονικό διάστημα ρυθμιστεί στην εντολή. Ο χρόνος αυτός μετράται σε χιλιοστά του δευτερολέπτου.

Αναφορικά με την PWM παλμοδότηση, στην ουσία αποτελεί μία περιοδική κυματομορφή με δύο τμήματα. Το τμήμα ON στο οποίο η κυματομορφή έχει τη μέγιστη τιμή και στο τμήμα OFF στο οποίο έχει την τιμή μηδέν. Το τμήμα ON ονομάζεται Duty Cycle και μετριέται είτε σε μονάδες χρόνου, όπως ms,us, είτε σε ποσοστό (%) επί της περιόδου. Εφαρμόζοντας μία PWM κυματομορφή στην τροφοδοσία ενός φορτίου επιτυγχάνουμε να ελέγξουμε το ποσοστό της ισχύος που πέφτει πάνω στο φορτίο. Αν για παράδειγμα το φορτίο είναι ένας κινητήρας, τότε επιτυγχάνουμε έλεγχο των στροφών του κινητήρα.[19]



Εικόνα 20: Ανάλυση της PWM παλμοδότησης

Πηγή : PWM παλμοδότηση

2.3 Προγραμματισμός Arduino

2.3.1 Γλώσσες χαμηλού επιπέδου

Η γλώσσα προγραμματισμού χαμηλού επιπέδου, είναι μία γλώσσα η οποία είναι πολύ κοντά στη γλώσσα μηχανής. Δε χρειάζεται διερμηνευτή ή μεταγλώττιση, γιατί το πρόγραμμα μπορεί να <<τρέξει>> στο συγκεκριμένο επεξεργαστή όπως είναι. Ο προγραμματιστής, έχει τον απόλυτο έλεγχο της συμπεριφοράς της μηχανής και μπορεί να επιτύχει διάφορους χρονισμούς με μεγάλη ακρίβεια. Ωστόσο, η εκμάθηση της γλώσσας χαμηλού επιπέδου απαιτεί μεγαλύτερο κόπο καθώς διαφέρει από μικροελεγκτή κάθε φορά. Ακόμη, τα προγράμματα γίνονται πολύπλοκα με συνέπεια σε περίπτωση λάθους να είναι δύσκολος ο εντοπισμός και η διόρθωσή του.[20]

2.3.2 Γλώσσες υψηλού επιπέδου

Ως γλώσσα προγραμματισμού υψηλού επιπέδου, ορίζεται η γλώσσα η οποία επιτρέπει τη μεταφορά ενός προγράμματος από έναν υπολογιστή σε έναν άλλο. Αποτελείται από εντολές, οι οποίες κατανοούνται εύκολα από τον προγραμματιστή και απαιτεί τη χρήση μεταγλωττιστή για τη μετατροπή του προγράμματος σε γλώσσα μηχανής. Οι γλώσσες υψηλού επιπέδου γενικής χρήσεως, είναι η Python, η C, η Visual Basic και η Java, ενώ οι γλώσσες ειδικού σκοπού είναι η Fortran, η Cobol και η Lisp.[21]

2.3.3 Καταχώρηση βιβλιοθηκών

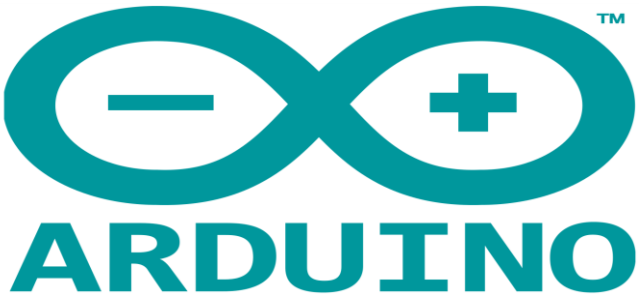
Για τη σωστή λειτουργία των αισθητήρων της εργασίας χρησιμοποιήθηκαν ορισμένες βιβλιοθήκες οι οποίες δηλώνονται πάντα στην αρχή του προγράμματος.

1. ThingSpeak : Είναι η βιβλιοθήκη για την επικοινωνία της πλατφόρμας ThingSpeak με τον κώδικα.
2. DHT : Με την παραπάνω βιβλιοθήκη γίνεται η αναγνώριση του αισθητήρα θερμοκρασίας - υγρασίας.
3. WiFi Client : Με την εντολή αυτή το NodeMCU μπορεί να συνδεθεί στην IP του δικτύου WiFi.
4. ESP8266 : Η βιβλιοθήκη αυτή διαθέτει ενσωματωμένο πρωτόκολλο επικοινωνίας και επιτρέπει σε οποιονδήποτε μικροεπεξεργαστή να συνδεθεί στο δίκτυο.
5. Wire : Με τη συγκεκριμένη βιβλιοθήκη μπορούν να επικοινωνήσουν όσες συσκευές διαθέτουν το ίδιο πρωτόκολλο επικοινωνίας, δηλαδή το I2C.

ΚΕΦΑΛΑΙΟ 3ο

3.1 Πλατφόρμα Arduino

3.1.1 Η έννοια του Arduino



Εικόνα 21: Λογότυπο εταιρείας Arduino

Πηγή : Arduino Wikipedia

Το Arduino είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα ανοιχτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες ,υλοποιημένες επίσης στη C++). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες. Το διάγραμμα και οι πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για εκείνους που θέλουν να το συναρμολογήσουν μόνοι τους κι αυτό είναι το κύριο χαρακτηριστικό που το διαφοροποιεί. Μπορεί να κατασκευαστεί από τον καθένα και συγχρόνως να ενσωματωθεί σε συσκευές για εμπορικούς σκοπούς. Με αυτό ενασχολείται μια ολόκληρη κοινότητα και κάτι τέτοιο έχει σα συνέπεια να υπάρχει διαθέσιμος τεράστιος όγκος ελεύθερης πληροφορίας. Τα Projects μπορούν να είναι αυτόνομα σε επίπεδο hardware αλλά και να επικοινωνούν με κάποιο λογισμικό (software) του Η/Υ του προγραμματιστή. Ακόμη, το Arduino πλέον χρησιμοποιεί ένα ειδικά προγραμματιζόμενο Atmega382 αντί του chip FTDI, με αποτέλεσμα να επιτρέπει την ταχύτερη μεταφορά των δεδομένων και της σειριακής επικοινωνίας. Ο μικροεπεξεργαστής ενός Arduino συνήθως προγραμματίζεται εκ των προτέρων , ώστε να έχει κάποιο φορτωτή εκκίνησης(BootLoader).

Ουσιαστικά, ο φορτωτής εκκίνησης, απλοποιεί τη διαδικασία της αποθήκευσης των προγραμμάτων στη flash memory του Arduino μέσω της σειριακής θύρας USB.[22]

3.1.2 Ιστορική Αναδρομή

Το Arduino ξεκίνησε σαν ένα σχέδιο το 2005 προκειμένου να δημιουργηθεί μία συσκευή για τον έλεγχο προγραμμάτων διαδραστικών σχεδίων από μαθητές , η οποία θα ήταν πιο οικονομική σε σχέση με άλλα παρόμοια συστήματα που ήταν διαθέσιμα εκείνη την περίοδο. Από το Μάιο του 2011, υπήρχαν διαθέσιμα περισσότερα από 300.000 Arduino. Οι εφευρέτες, Massimo Banzi και David Cueartielles, ονόμασαν το έργο τους "Arduin of Ivrea".Το Arduino είναι επίσης ένα ιταλικό όνομα που σημαίνει <<γενναίος φίλος>>.

Ο Κολομβιανός καλλιτέχνης και προγραμματιστής Hernando Barragan, δημιούργησε συρμάτωση "Wiring" ως μια μεταπτυχιακή διπλωματική του εργασία στο Interaction Design Institute Ivrea υπό την εποπτεία του Massimo Banzi και του Casey Reas. Η καλωδίωση "Wiring" βασίστηκε στην επεξεργασία και το ολοκληρωμένο περιβάλλον ανάπτυξης που είχε δημιουργηθεί από τον Casey Reas και τον Ben Fry. Η καλωδίωση , που ήταν η εργασία του Hernando , επρόκειτο να είναι μια ηλεκτρονική έκδοση της επεξεργασίας που χρησιμοποιήθηκε σε προγραμματιστικό περιβάλλον και ήταν η βάση για τη σύνταξη επεξεργασίας. Επόπτες αυτού, ήταν ο Hernando και ο Massimo Banzi, ιδρυτές ενός Arduino.

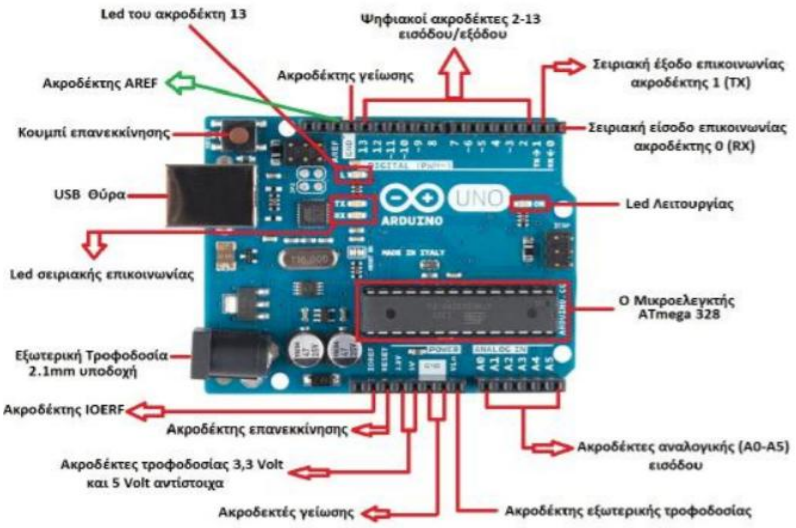
Μια σημαντική πτυχή του Arduino, που αποτελείται από έναν 8-bit μικροεπεξεργαστή της Atmel AVR, είναι ότι επιτρέπει στη CPU να συνδεθεί με μία ποικιλία από πρόσθετες λειτουργικές μονάδες, οι οποίες ονομάζονται "shields". Αυτές οι μονάδες, επικοινωνούν με την πλακέτα του Arduino άμεσα μέσω διαφόρων ακίδων. Ωστόσο, πολλές μονάδες είναι μεμονωμένα καθορισμένες μέσω ενός σειριακού διαύλου I^2C, ο οποίος επιτρέπει σε παραπάνω από μία μονάδες να λειτουργούν παράλληλα. Το επίσημο Arduino, χρησιμοποίησε τη σειρά megaAVR και ειδικά τους ATmega8, ATmega168, ATmega328, ATmega1280 και ATmega2560. Οι περισσότερες πλακέτες, περιλαμβάνουν μια γραμμική ρύθμιση 5 volt και μία 16 MHz crystal oscillato(ή κεραμικό συντονιστή σε ορισμένες παραλλαγές), αν και μερικά σχέδια όπως το LilyPad εκτελείται στα 8 MHz και βασίζεται από τους ρυθμιστές τάσης της πλακέτας λόγω ειδικού παράγοντα με περιορισμούς.

Εννοιολογικά, όταν χρησιμοποιείται λογισμικό στοίβας Arduino, όλες οι μονάδες "shields" προγραμματίζονται μέσω μίας σειριακής σύνδεσης RS-232, αλλά ο τρόπος που αυτό υλοποιείται διαφέρει ανάλογα με την έκδοση υλικού. Οι σειριακές πλάκες Arduino,

περιλαμβάνουν ένα απλό κύκλωμα για τη μετατροπή του σήματος επιπέδου RS-232 σε TTL. Τα σημερινά Arduino, προγραμματίζονται μέσω USB.[22]

3.1.3 Τεχνικά χαρακτηριστικά Arduino Uno

Όπως αναφέρθηκε και παραπάνω, η πιο διαδεδομένη πλακέτα Arduino είναι το Arduino Uno. Πολλές πλακέτες έχουν παρόμοια χαρακτηριστικά με κάποιες μικροδιαφορές αναλόγως με το πού θέλουμε να χρησιμοποιήσουμε την κάθε μία από αυτές. Το Arduino βασίζεται στο μικροελεγκτή ATmega 328 της Atmel. Πρόκειται για έναν 8-bit RISC μικροελεγκτή που είναι χρονισμένος στα 16MHz και διαθέτει ενσωματωμένη μνήμη. Η σύνδεση του στον υπολογιστή γίνεται μέσω της θύρας USB που διαθέτει και χρησιμοποιείται είτε για τον προγραμματισμό του είτε για την τροφοδοσία της πλακέτας. Στο επάνω μέρος της υπάρχουν 14 θηλυκές υποδοχές (pins 2-13) τα οποία λειτουργούν σαν ψηφιακές είσοδοι ή έξοδοι (I / O). Τα 6 pins με το σύμβολο '~' , μπορούν να χρησιμοποιηθούν ως PWM έξοδοι. Το καθένα από αυτά μπορεί να παρέχει το πολύ 40mA και λειτουργούν με τάση 5 Volt. Εφαρμόζοντας τάση 3.3 Volt σε μία είσοδο-pin, τότε αυτό μπορεί να παρέχει DC ρεύμα των 50 mA. Στο κάτω μέρος της πλακέτας μπορούμε να δούμε μία ακόμα σειρά από 6 pin με την ένδειξη ANALOG IN(A0-A5). Τα συγκεκριμένα pins χρησιμοποιούνται ως αναλογικοί είσοδοι κάνοντας χρήση του ADC (Analog to Digital Converter) που είναι ενσωματωμένο στο μικροελεγκτή.[23,24]



Εικόνα 22: Ανάλυση θυρών της πλακέτας Arduino Uno

Πηγή : Ανάλυση Arduino

Δίπλα ακριβώς από αυτά τα pins βρίσκονται ακόμη 6 με την ένδειξη POWER. Το καθένα από αυτά υπάρχει με σκοπό να εκτελεί κάποια συγκεκριμένη διαδικασία ή να τροφοδοτεί εξαρτήματα με συγκεκριμένη τάση το καθένα. Η τροφοδοσία του Arduino που συνίσταται, είναι από 7-12 Volt και μπορεί να προέρχεται από έναν κοινό μετασχηματιστή, μπαταρίες ή οποιαδήποτε άλλη πηγή DC αλλά και μέσω της θύρας USB που διαθέτει και συνδέεται κατευθείαν με τον υπολογιστή. Δίπλα από τους ακροδέκτες της τροφοδοσίας είναι τοποθετημένες οι γειώσεις, οι οποίες παρουσιάζονται με την ένδειξη GND. Η ένδειξη AREF στο επάνω μέρος της πλακέτας αναφέρεται σε έναν αναλογικό-ψηφιακό μετατροπέα αναφοράς τάσης, της τάσης εισόδου. Ο ακροδέκτης IOREF παρουσιάζει την τάση στις εισόδους/εξόδους (I/O) αυτής της πλακέτας. Επάνω στην πλακέτα μπορούμε να δούμε επίσης ένα κουμπί επανεκκίνησης και 4 μικροσκοπικά LED. Το LED με την ένδειξη On μας ενημερώνει για το αν βρίσκεται σε λειτουργία ή όχι η πλακέτα. Τα 2 LED με τις ενδείξεις TX και RX χρησιμοποιούνται ως ένδειξη λειτουργίας και ανάβουν όταν το Arduino στέλνει ή λαμβάνει δεδομένα μέσω του USB. Το 4ο LED ενσωματώθηκε από τους κατασκευαστές προκειμένου να μπορεί να γίνει οποιαδήποτε δοκιμή λειτουργίας μέσω του pin 13. Η μνήμη που διαθέτει η συγκεκριμένη πλακέτα είναι στα 32 KB για τη μνήμη Flash, με τα 0.5 KB να είναι δεσμευμένα από τον φορτωτή εκκίνησης (Bootloader), στα 2 KB για τη μνήμη SRAM και στα 1 KB για τη μνήμη EEPROM. Τέλος, η πλακέτα έχει διαστάσεις 68.6 mm x 53.4 mm με το βάρος της να είναι 25 g. Συνοπτικά τα χαρακτηριστικά του Arduino Uno παρουσιάζονται στον παρακάτω πίνακα.

ARDUINO UNO	
Μικροελεγκτής	ATmega 328
Τάση λειτουργίας	5 Volt
Τάση εισόδου	7-12 Volt
Ψηφιακά I/O Pins	14 (2-13)
Αναλογικές είσοδοι	6 (A0-A5)
PWM είσοδοι	6 (3,5,6,9,10,11)
DC ρεύμα ανά I/O	40 mA

DC ρεύμα για 3.3 Volt	50 mA
Γειώσεις(GND)	3
Μνήμη Flash	32 KB
Μνήμη SRAM	2 KB
Μνήμη EEPROM	1 KB
Ταχύτητα (Clock Speed)	16 MHz
Συνδεσιμότητα με Η/Υ	USB Καλώδιο
Διαστάσεις	68.6 mm x 54.3 mm
Βάρος πλακέτας	25 g

Πίνακας 3: Τεχνικά Χαρακτηριστικά Arduino Uno

3.1.4 Ιστορία του NodeMCU

Το NodeMCU, δημιουργήθηκε λίγο μετά την εμφάνιση του ESP8266 Wi-Fi Micro-Chip. Στις 30 Δεκεμβρίου του 2013 η εταιρεία Espressif Systems ξεκίνησε την παραγωγή του ESP8266 Micro-Chip, ενώ η διαδικασία παραγωγής του NodeMCU ξεκίνησε στις 15 Οκτωβρίου του 2015, όταν ο Χονγκ διέθεσε το πρώτο αρχείο υλικολογισμικού στο GitHub. Δύο μήνες αργότερα, το project επεκτάθηκε ώστε να περιλαμβάνει μία πλατφόρμα ανοιχτού υλικού, όταν ο προγραμματιστής Huang R διέθεσε το αρχείο τυπωμένου κυκλώματος - PCB, gerber ενός ESP8266 πίνακα, με την ονομασία devkit v0.9. Αργότερα τον ίδιο μήνα, ο Tuan PM , μετέφερε τη βιβλιοθήκη πελατών MQTT(Message Queuing Telemetry Transport) από το Contiki(Operating System for low-power Internet of Things Devices) στην πλατφόρμα του ESP8266. Τότε, το NodeMCU ήταν ικανό να υποστηρίξει το MQTT IoT Protocol, χρησιμοποιώντας τη γλώσσα προγραμματισμού Lua για την πρόσβαση στο MQTT. Ύστερα από κάποιες προσθήκες που ακολούθησαν, το καλοκαίρι του 2016 το NodeMCU περιείχε πάνω από 40 διαφορετικές ενότητες.[25]

3.1.5 Προέλευση μονής πλακέτας

Οι μικροελεγκτές μονής πλακέτας πρωτοεμφανίστηκαν στα τέλη της δεκαετίας του 1970, όταν με την εμφάνιση των πρώτων μικροεπεξεργαστών, όπως του 6502 MOS Technology και του Z80 Zilog κατάφεραν να υλοποιήσουν έναν ολόκληρο ελεγκτή σε πλακέτα, με σκοπό τη δημιουργία ενός υπολογιστή για την επίτευξη μιας μικρής εργασίας. Το Μάρτιο του 1976, η Intel ανακοίνωσε έναν υπολογιστή μονής πλακέτας ο οποίος περιείχε όλα τα απαραίτητα υλικά υποστήριξης για το μικροεπεξεργαστή 8080, ο οποίος είχε πρωτοεμφανιστεί τον Απρίλιο του 1974. Ο υπολογιστής αυτός, συνοδευόταν με 1 kilobyte RAM, 4 kilobytes ROM για τον προγραμματισμό από το χρήστη και 48 παράλληλες γραμμές από ψηφιακές εισόδους-εξόδους (I/O) μαζί με τους οδηγούς γραμμής για την επικοινωνία των ψηφιακών σημάτων με την πλακέτα και τα καλώδια. Ακόμη, διέθετε και την επιλογή για επέκταση του υλικού - hardware σε περιπτώσεις εφαρμογών που χρειαζόταν. Η ανάπτυξη λογισμικού για αυτό το σύστημα φιλοξενήθηκε στο σύστημα ανάπτυξης της Intellec MDS, το οποίο παρείχε υποστήριξη συναρμολόγησης, PL/M (Programming Language for Microcomputer) καθώς και εξομοίωση για τον εντοπισμο και επίλυση τυχόν σφαλμάτων. Οι επεξεργαστές αυτή της γενιάς, απαιτούσαν την ύπαρξη περαιτέρω chips πέρα από αυτό του επεξεργαστή. Η μνήμη RAM και η μνήμη EPROM ήταν ξεχωριστά και συχνά απαιτούσαν διαχείριση μνήμης ή κύκλωμα ανανέωσης για δυναμική μνήμη.[26]

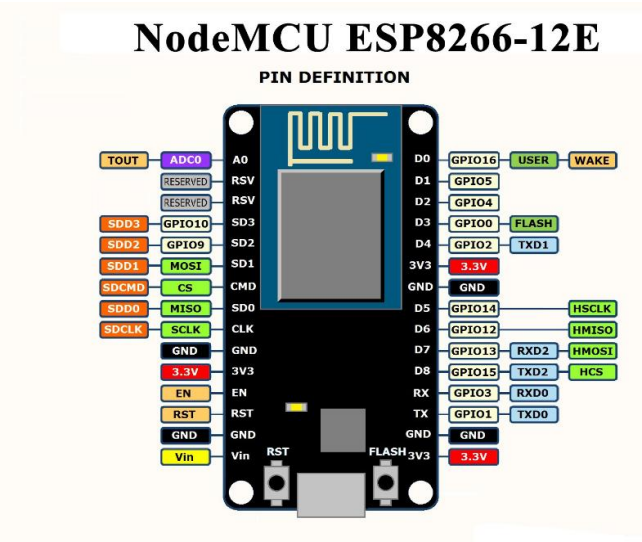
3.1.6 Τεχνικά χαρακτηριστικά NodeMCU

Η τροφοδοσία του NodeMCU γίνεται μέσω μια θύρας Micro-USB που διαθέτει η πλακέτα ή μιας ρυθμισμένης τάσης στα 3,3 Volt ή μέσω μιας εξωτερικής πηγής τροφοδοσίας 7-12 Volt στους ακροδέκτες Vin. Η γείωση βρίσκεται στα pins GND της πλακέτας.

- Διαθέτει 17 εισόδους-εξόδους (I/O) γενικού σκοπού. Απ΄αυτές 1 είναι αναλογική στο pin A0. Οι 16 ψηφιακές, είναι στα pins D0-D8, SD2, SD3, RX, TX - GPIO 0-16 και υποστηρίζει PWM παλμοδότηση στα pins D1-D4, D6-D8 ,RSV.
- Έχει ενσωματωμένους 4 ακροδέκτες για επικοινωνία τύπου SPI(SD1,CMD,SD0 CLK).
- Διαθέτει πρωτόκολλο επικοινωνίας UART, UART0(RXD0 & TXD0) και UART1(RXD1 & TXD1) με το UART1 να χρησιμοποιείται να το ανέβασμα του προγράμματος.

- Στον τομέα της μνήμης, το NodeMCU διαθέτει 4 MB flash memory και 64 KB SRAM.
- Το ρολόι του είναι χρονισμένο στα 80 MHz και έχει ενσωματωμένη κεραία.

Σχηματικά, η πλακέτα του NodeMCU, παρουσιάζεται στην παρακάτω εικόνα.[27,28]



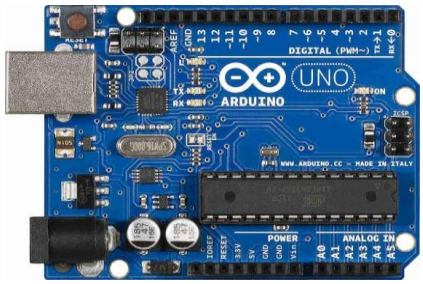
Εικόνα 23: Σχηματική αναπαράσταση του NodeMCU ESP8266-12E

Πηγή : Σχήμα NodeMCU

3.1.7 Εκδόσεις Arduino

Παρακάτω, αναλύονται μερικές από τις βασικότερες εκδόσεις του Arduino. Η κάθε μία είναι διαφορετική και η επιλογή της εξαρτάται από την εφαρμογή που επιλέγεται να κατασκευαστεί κάθε φορά.

Arduino Uno

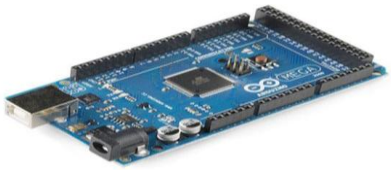


Εικόνα 24: Πλακέτα Arduino Uno

Πηγή : Arduino Uno

Είναι η πιο διαδεδομένη πλακέτα με αρκετά χαμηλό κόστος και χρησιμοποιείται για απλές εφαρμογές. Χρησιμοποιεί την τεχνολογία ATmega328. Περιλαμβάνει 14 εισόδους/εξόδους (I/O), εκ των οποίων 6 ως PWM έξοδοι και 6 αναλογικές εισόδους. Είναι χρονισμένο στα 16 MHz και έχει τάση λειτουργίας τα 5 Volt.[29]

Arduino Mega



Εικόνα 25: Πλακέτα Arduino Mega

Πηγή : Arduino Mega

Το Arduino Mega χρησιμοποιεί τεχνολογία Surface-mounted ATmega1280. Τα πλεονεκτήματά του είναι η ολική μνήμη του στα 256kb , καθώς και οι 54 ψηφιακές ακίδες εισόδου/εξόδου , οι 12 αναλογικές είσοδοι και το ρολόι των 16 MHz. Η τάση λειτουργίας , είναι τα 5 Volt.[30]

ArduinoADK

Εικόνα 26: Πλακέτα ArduinoADK

Πηγή : Arduino ADK

Η συγκεκριμένη πλακέτα έχει τα ίδια χαρακτηριστικά με το Arduino Mega που έχει αναφερθεί προηγουμένως ,όμως διαθέτει επιπλέον μία θύρα USB για σύνδεση με τα τηλέφωνα Android, το οποίο βασίζεται στο MAX3421e IC.[31]

ArduinoLeonardo

Εικόνα 27: Πλακέτα Arduino Leonardo

Πηγή : Arduino Leonardo

Το Arduino Leonardo είναι μια πλακέτα η οποία με την εισαγωγή Altmega 32U4 chip μπορεί να χρησιμοποιηθεί σαν ψηφιακό πληκτρολόγιο ή ποντίκι και εξαλείφει την ανάγκη για συνδεσιμότητα μέσω της θύρας USB. Διαθέτει 20 ψηφιακές εισόδους/εξόδους (I/O), εκ των οποίων 7 μπορούν να χρησιμοποιηθούν ως PWM έξοδοι και 12 αναλογικές εισόδους. Είναι χρονισμένο στα 16 MHz και η τάση του είναι τα 5 Volt.[32]

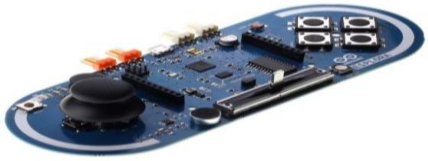
ArduinoDue

Εικόνα 28: Πλακέτα Arduino Due

Πηγή : Arduino Due

Είναι η πρώτη πλακέτα που βασίζεται σε έναν 32-bit πυρήνα ARM μικροελεγκτή και διαθέτει 54 ψηφιακές ακίδες εισόδου/εξόδου (I/O), 12 αναλογικές εισόδους και 2 DAC (digital-to analog-converter) και 2 CAN . Η τάση λειτουργίας στις ακίδες είναι 3,3 Volt και είναι χρονισμένο στα 84 MHz.[33]

ArduinoEsplora



Εικόνα 29: Πλακέτα Arduino Esplora

Πηγή : Arduino Esplora

Η έκδοση αυτή, βασίζεται στην έκδοση Leonardo και περιέχει ενσωματωμένους αισθητήρες με εμφάνιση που παραπέμπει σε χειριστήριο κονσόλας βιντεοπαιχνιδιών. Περιλαμβάνει εισόδους-εξόδους ήχου και φωτός, ποτενσιόμετρο, αισθητήρα θερμοκρασίας, μικρόφωνο, καθώς και επιταχυνσιόμετρο.Ακόμη, περιλαμβάνει ειδική υποδοχή για τη διασύνδεση του με οθόνη TFT LCD. Η πλακέτα αυτή, είναι χρονισμένη στα 16 MHz και λειτουργεί με τάση 5 Volt.[34]

ArduinoEthernet

Εικόνα 30: Arduino Ethernet

Πηγή : Arduino Ethernet

Η πλακέτα αυτή χρησιμοποιεί το μικροελεγκτή ATmega238. Διαθέτει 15 ψηφιακές εισόδους/εξόδους, 6 αναλογικές εισόδους και είναι χρονισμένο στα 16 MHz. Ακόμη, επιτρέπει τη σύνδεση με το διαδίκτυο και η τροφοδοσία αυτού γίνεται μέσω καλωδίου Ethernet. Επίσης διαθέτει υποδοχή για κάρτα micro SD.[35]

ArduinoYun

Εικόνα 31: Arduino Yun

Πηγή : Arduino Yun

Η συγκεκριμένη πλακέτα είναι βασισμένη στο μικροελεγκτή ATmega32u4 και στον ATheros AR9331. Διαθέτει υποδοχή RJ-45 για σύνδεση στο διαδίκτυο με καλώδιο, αλλά και ασύρματα μέσω Wifi. Υποστηρίζει 20 ψηφιακές εισόδους/εξόδους(I/O), 7 εκ των οποίων μπορούν να χρησιμοποιηθούν ως PWM έξοδοι και 12 αναλογικές εισόδους. Το ρολόι της συγκεκριμένης πλακέτας είναι χρονισμένο στα 16 MHz, διαθέτει υποδοχή για κάρτα μνήμης SD και η τάση λειτουργίας του είναι τα 5 Volt. Είναι ένας συνδυασμός Arduino και Linux.[36]

ArduinoBluetooth

Εικόνα 32: Arduino Bluetooth

Πηγή : Arduino Bluetooth

Είναι βασισμένο στο μικροελεγκτή ATmega 328 P. Υποστηρίζει ασύρματη σειριακή επικοινωνία μέσω Bluetooth, χωρίς όμως να είναι συμβατό με Bluetooth, ακουστικά και άλλες συσκευές ήχου. Διαθέτει 14 ψηφιακές εισόδους/εξόδους (I/O), 6 εκ των οποίων μπορούν να χρησιμοποιηθούν ως PWM έξοδοι, μία για Reset της πλακέτας και 6 αναλογικές εισόδους. Το ρολόι της πλακέτας είναι χρονισμένο στα 16 MHz, με τάση λειτουργίας τα 5 Volt.[37]

ArduinoPro

Εικόνα 33: Arduino Pro

Πηγή : Arduino Pro

Η παραπάνω πλακέτα είναι βασισμένη στο μικροελεγκτή ATmega328 .Υπάρχουν 2 εκδόσεις της παραπάνω μορφής, μία των 3,3 Volt τάσης λειτουργίας στα 8 MHz και μία των 5 Volt στα 16 MHz . Διαθέτει 14 ψηφιακές εισόδους/εξόδους (I/O), 6 εκ των οποίων μπορούν να χρησιμοποιηθούν ως PWM έξοδοι και 6 αναλογικές εισόδους. Είναι μικρό σε μέγεθος και χωρίς ακίδες για να υπάρχει η δυνατότητα προσαρμογής σε μικρά έργα.[38]

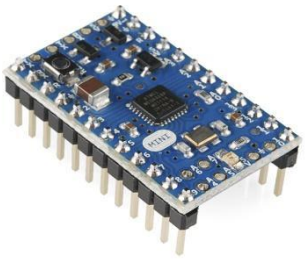
ArduinoFio

Εικόνα 34: Arduino Fio

Πηγή : Arduino Fio

Η πλακέτα αυτή, βασίζεται στο μικροελεγκτή ATmega328P. Διαθέτει 14 ψηφιακές εισόδους/εξόδους(I/O), 6 εκ των οποίων μπορούν να χρησιμοποιηθούν ως PWM έξοδοι, 8 αναλογικές εισόδους. Δέχεται συνδέσεις με μπαταρίες Lithium και περιλαμβάνει σύστημα φόρτισης. Προορίζεται καθαρά για εφαρμογές με ασύρματη επικοινωνία. Τα δεδομένα φορτώνονται μέσω καλωδίου FTDI. Η τάση λειτουργίας του είναι στα 3,3 Volt και το ρολόι του είναι χρονισμένο στα 8 MHz.[39]

ArduinoMini



Εικόνα 35: Arduino Mini

Πηγή : Arduino Mini

Το Arduino Mini είναι βασισμένο στην έκδοση Arduino Nano και στο μικροελεγκτή ATmega328. Διαθέτει 14 ψηφιακές εισόδους/εξόδους(I/O), 6 εκ των οποίων μπορούν να χρησιμοποιηθούν ως PWM έξοδοι και 8 αναλογικές εισόδους.Είναι χρονισμένο στα 16 MHz, δε διαθέτει θύρα USB για τη σειριακή επικοινωνία και η τροφοδοσία του γίνεται με 5 Volt.[40]

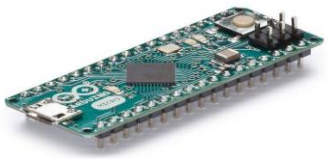
Arduino Duemilanove

Εικόνα 36: Arduino Duemilanove

Πηγή : Arduino Duemilanove

Το Arduino Duemilanove είναι βασισμένο στο μικροελεγκτή ATmega168 ή στον ATmega328. Διαθέτει 14 ψηφιακές εισόδους/εξόδους(I/O),εκ των οποίων 6 μπορούν να χρησιμοποιηθούν ως έξοδοι σε PWM παλμοδότηση και 6 αναλογικές εισόδους. Το ρολόι χρονισμού του, είναι ρυθμισμένο στα 16 MHz και η τάση λειτουργίας του, είναι τα 5 Volt.[41]

Arduino Micro



Εικόνα 37: Arduino Micro

Πηγή : Arduino Micro

Το Arduino Micro, βασίζεται στο μικροελεγκτή ATmga23U4. Διαθέτει 20 ψηφιακές εισόδους/εξόδους(I/O), εκ των οποίων 7 μπορούν να χρησιμοποιηθούν ως έξοδοι σε PWM παλμοδότηση και 12 ως αναλογικές εισόδους. Το ρολόι του είναι χρονισμένο στα 16 MHz, ενώ αντί για θύρα USB που διαθέτουν οι υπόλοιπες πλακέτες Arduino, αυτή διαθέτει Micro USB. Η τάση λειτουργίας της , είναι τα 5 Volt.[42]

Arduino Robot

Εικόνα 38: Arduino Robot

Πηγή : Arduino Robot

Το Arduino Robot είναι το πρώτο Arduino που κατασκευάστηκε πάνω σε ρόδες. Διαθέτει 2 μικροεπεξεργαστές, έναν σε κάθε πλακέτα του για τον έλεγχο του μοτέρ για τις ρόδες, αλλά και για τον έλεγχο των αισθητήρων. Οι 2 αυτοί μικροεπεξεργαστές, βασίζονται στον ATmega32U4. Έχει ενσωματωμένους αισθητήρες και ο προγραμματισμός του γίνεται όπως και στην περίπτωση του Leonardo.[43]

Για το Arduino, πέρα από κάποιες βασικές πλακέτες που προαναφέρθηκαν, αξίζει να σχολιαστούν και η ποικιλία των διαθέσιμων shields που κυκλοφορούν στο εμπόριο. Αυτά έχουν ως αποτέλεσμα να μετατρέπουν την “απλή” πλακέτα Arduino σε πιο σύνθετη, δίνοντάς της περισσότερες επιλογές και χαρακτηριστικά. Μερικά παραδείγματα, παρουσιάζονται παρακάτω.

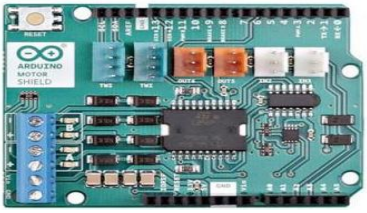
Arduino WiFi Shield

Εικόνα 39: Arduino WiFi Shield

Πηγή : Arduino WiFi Shield

Η συγκεκριμένη επέκταση, επιτρέπει στο Arduino να συνδεθεί στο δίκτυο ασύρματα, μέσω απλών βημάτων εγκατάστασης.[44]

Arduino Motor Shield

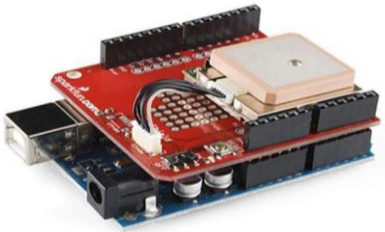


Εικόνα 40: Arduino Motor Shield

Πηγή : Arduino Motor Shield

Η παραπάνω πλακέτα, διαθέτει ένα πλήρες διπλό οδηγό, το οποίο μπορεί να διαχειριστεί επαγωγικά φορτία όπως ρελέ, DC και βηματικούς κινητήρες.[45]

Arduino GPS Shield



Εικόνα 41: Arduino GPS Shield

Πηγή : Arduino GPS Shield

Το GPS shield, επιτρέπει την εύρεση της τοποθεσίας της συσκευής χωρίς να επηρεάζει σημαντικά την πηγή τροφοδοσίας της συσκευής. Δέχεται και επεξεργάζεται τα σήματα από το GPS, την πυξίδα και το Galileo(Global Navigation Satellite System).[46]

Arduino LCD Shield

Εικόνα 42: Arduino LCD Shield

Πηγή : Arduino LCD Shield

Αυτή η πλακέτα, έχει ενσωματωμένη οθόνη και πληκτρολόγιο και αναφέρεται σε εφαρμογές που χρειάζεται προβολή δεδομένων τοπικά. Διαθέτει επίσης κουμπιά για εύκολη περιήγηση κατά τη χρήση της.[47]

3.1.8 Δυνατότητες και Πλεονεκτήματα Arduino

Το Arduino αποτελεί ένα αρκετά χρήσιμο εργαλείο διότι επιτρέπει την κατασκευή ενός υπολογιστικού συστήματος το οποίο θα δίνει τη δυνατότητα να ελεγχθούν οι συσκευές του φυσικού κόσμου και να ληφθούν πληροφορίες. Σε αντίθεση με τον Ηλεκτρονικό Υπολογιστή, το περιβάλλον και το λογισμικό του είναι πολύ φιλικά και εύχρηστα ακόμα και σε νέους χρήστες. Μέσω του Arduino δίνεται η δυνατότητα να δημιουργηθούν συσκευές , οι οποίες θα μπορούν να δέχονται ερεθίσματα από το περιβάλλον με τη βοήθεια των κατάλληλων αισθητήρων και να αντιδρούν ανάλογα με το πώς έχουν προγραμματιστεί, όπως επίσης και να εξυπηρετούν προσωπικές ανάγκες και σκοπούς.

Όλα τα παραπάνω βέβαια δεν παρουσιάζουν ιδιαίτερη πρωτοτυπία. Μέχρι σήμερα έχουν κατασκευαστεί και άλλες παρόμοιες πλατφόρμες που έχουν τη δυνατότητα να δώσουν τις ίδιες πληροφορίες και να υλοποιούν ακριβώς τα ίδια πράγματα. Αξίζει να σημειωθεί πως η πλατφόρμα Arduino έχει αρκετά πλεονεκτήματα, τα οποία την έχουν καταστήσει μία από τις πιο διαδεδομένες πλατφόρμες παγκοσμίως.[48]

Τα βασικά πλεονεκτήματα της πλατφόρμας Arduino είναι :

- **Χαμηλό κόστος:** Σε σύγκριση με άλλες πλατφόρμες μικροελεγκτών που κυκλοφορούν στο εμπόριο οι πλακέτες Arduino είναι αρκετά οικονομικές, αφού οποιοσδήποτε επιθυμεί μπορεί να τις αποκτήσει έναντι λίγων χρημάτων. Είναι αρχιτεκτονικά ανοιχτή, που σημαίνει ότι μπορεί ο καθένας να την αναπτύξει μόνος του. Επιπλέον, για κάποιους που διακατέχονται από όρεξη και φυσικά περίσσιο χρόνο υπάρχει και η δυνατότητα να αγοραστεί και μη συναρμολογήσιμη έκδοση, ώστε να είναι ακόμη πιο οικονομική.

- **Μεταφέρσιμη:** Το λογισμικό που χρησιμοποιείται στην πλακέτα μπορεί να τρέξει σε διάφορα λειτουργικά συστήματα, όπως Windows, Linux και Macintosh OS X, σε αντίθεση με τις περισσότερες πλακέτες του εμπορίου, που τρέχουν μόνο στο περιβάλλον των Windows.

- **Επεκτάσιμη:** Το υλικό και το λογισμικό της πλατφόρμας είναι ανοιχτά και ελεύθερα για όλους τους ενδιαφερόμενους. Αυτό σημαίνει ότι καθημερινά πολλοί υποστηρικτές του ελεύθερου λογισμικού αναπτύσσουν όλο και περισσότερες βιβλιοθήκες για την υποστήριξη της πλατφόρμας.

- **Απλό προγραμματιστικό περιβάλλον:** Το περιβάλλον προγραμματισμού του Arduino είναι ιδιαίτερα φιλικό και εύκολο στη χρήση για αρχάριους, αλλά ταυτόχρονα και αρκετά ευέλικτο για τους πιο προχωρημένους χρήστες. Συνεπώς, οι αρχάριοι χρήστες μπορούν πολύ εύκολα και γρήγορα να υλοποιήσουν την πρώτη τους κατασκευή και συγχρόνως οι προχωρημένοι να δημιουργήσουν μία πιο σύνθετη.

3.1.9 Κατηγορίες μικροελεγκτών

Όπως άλλωστε και σε όλα τα θέματα, έτσι κι εδώ ο ανταγωνισμός έχει επηρεάσει τον τομέα των μικροελεγκτών. Οι μικροελεγκτές, λόγω του ότι μπορούν να ενσωματωθούν σε κάθε ηλεκτρική και ηλεκτρονική συσκευή έχουν να αντιμετωπίσουν τεράστιο ανταγωνισμό από τις βιομηχανίες του χώρου τους. Οι κυριότερες κατηγορίες που έχουν ξεχωρίσει, είναι οι παρακάτω :

- Μικροελεγκτές των 8 bit συνήθως, αλλά και των 4 bit, πολύ χαμηλού κόστους που προορίζονται για γενική χρήση. Διαθέτουν μικρό αριθμό ακροδεκτών, καμιά φορά μπορεί και κάτω των 8. Ο σχεδιασμός τους, γίνεται με έμφαση στη χαμηλή κατανάλωση ισχύος και στην ιδιότητά τους να λειτουργούν αυτόνομα, χωρίς άλλα εξωτερικά εξαρτήματα. Δεν έχουν δυνατότητα επέκτασης της μνήμης τους και το εσωτερικό λογισμικό τους δεν είναι εύκολο να αντιγραφεί. Τα ευρέως γνωστά μοντέλα αυτών των μικροελεγκτών ,είναι της σειράς PIC(Microchip), AVR(Atmel) και 8051(Intel,Atmel,Dallas κα).

- Μικροελεγκτές των 12 ή 32 bit, αλλά και των 8 bit χαμηλού κόστους και γενικής χρήσης, οι οποίοι διαθέτουν σχετικά μεγάλο αριθμό ακροδεκτών. Επίσης, διαθέτουν μεγάλο αριθμό περιφερειακών, όπως θύρες UART, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικο. Στους κατασκευαστές της Άνω Ανατολής, συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Η μνήμη τους, μπορεί να αυξηθεί με εξωτερική επέκταση.

- Μικροελεγκτές κατα κύριο λόγο 32 bit και μέσου κόστους. Οι προηγούμενοι, προορίζονται για γενική χρήση και διαθέτουν μεγάλο αριθμό ακροδεκτών. Κύριο χαρακτηριστικό τους, είναι η ταχύτητα εκτέλεση των εντολών, η υψηλή αυτάρκεια περιφερειακών συσκευών και οι μεγάλες δυνατότητες εσωτερικής και εξωτερικής μνήμης προγράμματος Flash και Ram. Οι κατασκευαστές αυτών των μικροελεγκτών, δίνουν μεγάλη έμφαση στη μεταφερσιμότητα του λογισμικού.

- Τέλος, σε μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι συνήθως, διαθέτουν ένα προχωρημένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται σε hardware. Τέτοιοι μικροελεγκτές, χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές, όπως είναι τα μόντεμ.

Από όλα τα παραπάνω γίνεται αντιληπτό πως οι αυξημένες πωλήσεις εξακολουθούν να αναφέρονται στους μικροελεγκτές των 8 bit, καθώς έχουν το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού προσφέροντας το ίδιο αποτέλεσμα. Όλο αυτό σχετίζεται άμεσα με τη εξέλιξη της τεχνολογίας και τις βελτιωμένες επιδόσεις των σημερινών μικροελεγκτών σε σχέση με το παρελθόν.[49]

3.2 Αισθητήρες που χρησιμοποιήθηκαν

3.2.1 Αισθητήρες

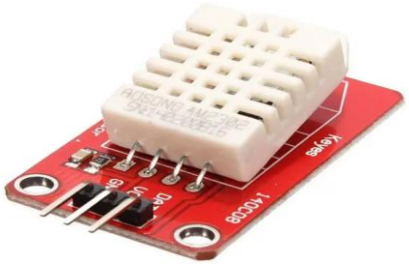
Ως αισθητήρας ορίζεται μία συσκευή η οποία ανιχνεύει ένα φυσικό μέγεθος και παράγει από αυτό μια μετρήσιμη έξοδο. Οι πρώτοι αισθητήρες και όργανα μέτρησης είναι μηχανικά. Η συστηματική μελέτη του ηλεκτρισμού, οδήγησε στην ανάπτυξη νέων αισθητήρων-ηλεκτρικών, η έξοδος των οποίων ήταν ένα αναλογικό σήμα. Ο τομέας των αισθητήρων εχει λάβει τεράστια εξέλιξη κι αυτό μπορεί να το συνειδητοποιήσει κάποιος σκεπτόμενος σαν παράδειγμα τα αυτοκίνητα της δεκαετίας του ‘60 και του ‘70. Περιλάμβαναν μόνο δύο απλούς ηλεκτρικούς αισθητήρες, έναν για τη μέτρηση της θερμοκρασίας του ψυκτικού υγρού και έναν δεύτερο για τη μέτρηση της στάθμης του καυσίμου, σε αντίθεση με τα σημερινά αυτοκίνητα τα οποία διαθέτουν πληθώρα αισθητήρων για κάθε μία λειτουργία ξεχωριστά. Λόγω της ανάγκης αντιμετώπισης προβλημάτων της σύγχρονης έρευνας και της τεχνολογίας, ο τομέας των αισθητήρων έχει αναπτυχθεί. Οι αισθητήρες χρησιμοποιούνται σε κάθε σχεδόν εφαρμογή και στόχο έχουν να εξυπηρετούν το

χρήστη. Χαρακτηρίζονται από αρκετά στοιχεία, με τα οποία κάποιος μπορεί να επιλέξει κατάλληλα εκείνον που επιθυμεί , ανάλογα με την εφαρμογή του. Τα στοιχεία αυτά παρουσιάζονται συνοπτικά παρακάτω :

- Εύρος, ονομάζεται το όριο στο οποίο η συσκευή λειτουργεί αξιόπιστα.
- Ακρίβεια, είναι η εγγύτητα της τιμής εξόδου προς την τιμή εισόδου.
- Σφάλμα, είναι η διαφορά ανάμεσα στη μετρούμενη τιμή και την πραγματική.
- Ανοχή, είναι το μέγιστο σφάλμα που μπορεί να δημιουργήσει ο αισθητήρας.
- Ευαισθησία, χαρακτηρίζεται η σχέση αλλαγής εξόδου προς την αλλαγή εισόδου, η οποία είναι ίση με τη διαφορά των τιμών εξόδου προς τη διαφορά των αντίστοιχων τιμών εισόδου.
- Βαθμονόμηση, είναι η βαθμολόγηση της κλίμακας σε μονάδες.
- Διακριτική ικανότητα, είναι η μικρότερη αλλαγή τιμής εισόδου που μπορεί να ανιχνεύσει.
- Νεκρή ζώνη, ορίζεται το μέγιστο ποσό αλλαγής της εισόδου που δεν επιφέρει αλλαγή στην έξοδο.
- Γραμμικότητα, είναι ο βαθμός στον οποίο η γραφική παράσταση της εξόδου προσεγγίζει ευθεία ως προς την είσοδο του αισθητήρα.
- Απόκριση, είναι ο χρόνος που απαιτείται για να λάβει την τελική τιμή η έξοδος.
- Καθυστέρηση, είναι η καθυστέρηση της αλλαγής εξόδου ως προς τη είσοδο.
- Ευστάθεια, είναι η μεταβολή της εξόδου σε μεγάλη χρονική περίοδο, χωρίς μεταβολή της εισόδου και των συνθηκών.
- Υστέρηση, είναι η διαφορά στην έξοδο όταν η κατεύθυνση της μεταβολής της εισόδου αντιστραφεί.
- Επαναληψιμότητα, είναι η παραγωγή του ιδίου αποτελέσματος σε διαφορετικές χρονικές στιγμές, με την ίδια είσοδο.
- Ολίσθηση, είναι η μεταβολή των χαρακτηριστικών του αισθητήρα με το χρόνο και το περιβάλλον.
- Στατικό σφάλμα, είναι το σταθερό σφάλμα σε όλο το εύρος λειτουργίας.
- Χρόνος λειτουργίας, είναι ο εκτιμώμενος χρόνος λειτουργίας στα πλαίσια των προδιαγραφών του.[50]

3.2.2 Αισθητήρας θερμοκρασίας - υγρασίας

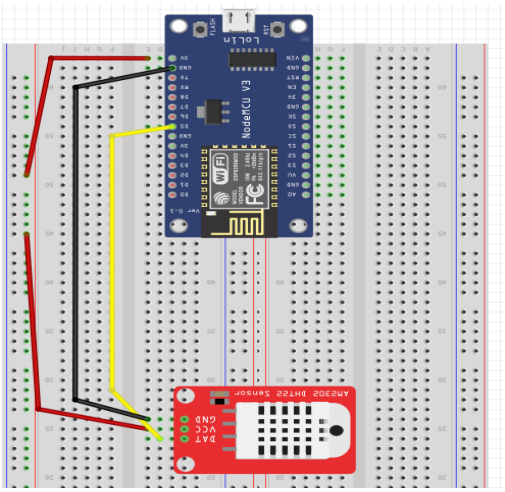
Ο αισθητήρας που χρησιμοποιήθηκε για αυτές τις μετρήσεις, είναι ο DHT22 που παρουσιάζεται σχηματικά στην παρακάτω εικόνα. Αποτελεί μία μονάδα με ψηφιακό σήμα εξόδου, η οποία περιέχει συγχρόνως βαθμονομημένο αισθητήρα θερμοκρασίας και υγρασίας. Χρησιμοποιεί τεχνολογία ανίχνευσης της θερμοκρασίας-υγρασίας, διασφαλίζοντας στα προϊόντα αξιοπιστία και μακροπρόθεσμη σταθερότητα. Διαθέτει ένα χωρητικό στοιχείο ανίχνευσης της υγρασίας και ένα θερμίστορ. Το θερμίστορ αποτελεί μία αντίσταση του οποίου η τιμή μεταβάλλεται με τη θερμοκρασία. Ακόμη, έχει έναν μικροελεγκτή των 8-bit με γρήγορη απόκριση και ποιότητα. Το μικρό του μέγεθος, η χαμηλή κατανάλωση ισχύος και η ικανότητά του για μέτρηση έως 20 μέτρα, τον καθιστούν εύχρηστο για πολλές εφαρμογές.



Εικόνα 43: Αισθητήρας θερμοκρασίας - υγρασίας

Πηγή : Αισθητήρας θερμοκρασίας - υγρασίας

Οι διαστάσεις του παραπάνω αισθητήρα είναι : 40 x 23 cm με βάρος 4g. Δέχεται τάση λειτουργίας 5 Volt και διαθέτει μία αμφίδρομη έξοδο. Η περιοχή θερμοκρασιών είναι από -40 έως 80°C ± 0.5 °C. Η υγρασία που μπορεί να μετρηθεί κυμαίνεται σε ποσοστό 20-90% RH ± 2% °C. Η διασύνδεση του αισθητήρα με το NodeMCU και η βιβλιοθήκη που χρησιμοποιήθηκε παρουσιάζονται παρακάτω.[51]

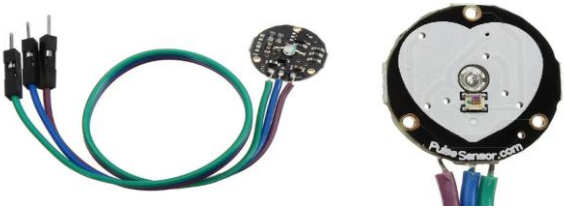


Εικόνα 44 : Διασύνδεση DHT 22 με NodeMCU

Το μαύρο καλώδιο συνδέεται από το Ground (GND) του αισθητήρα στο Ground (GND) του NodeMCU, το κόκκινο από τον ακροδέκτη των 3 Volt στο VCC του αισθητήρα και το κίτρινο καλώδιο από το pin DAT του αισθητήρα στο pin D5 του NodeMCU. Η βιβλιοθήκη που χρειάζεται αυτός ο αισθητήρας, είναι η “DHT.h” .

3.2.3 Αισθητήρας καρδιακών παλμών

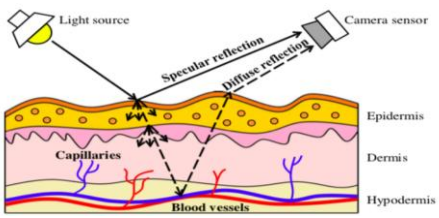
Ο αισθητήρας καρδιακών παλμών παρουσιάζεται στην εικόνα παρακάτω. Βασίζεται σε έναν απλό οπτικό αισθητήρα καρδιακού ρυθμού, ο οποίος συγχρόνως περιέχει κυκλώματα ενίσχυσης και ακύρωσης του θορύβου. Με αυτόν τον τρόπο, πραγματοποιείται εύκολη και γρήγορη λήψη του καρδιακού ρυθμού με αξιοπιστία. Η ενέργεια που καταναλώνει είναι 4mA γεγονός που την καθιστά ιδανική για κινητές εφαρμογές. Για τη χρήση της, αρκεί να είναι σε επαφή με το λοβό του αυτιού ή με το άκρο του δακτύλου του χρήστη. Το καλώδιο που τον συνοδεύει έχει σαν άκρα αρσενικές επαφές, επομένως δεν απαιτείται συγκόλληση.



Εικόνα 45: Αισθητήρας καρδιακών παλμών

Πηγή : Αισθητήρας καρδιακών παλμών

Η τυπική τάση λειτουργίας του αισθητήρα είναι τα 3.3 Volt DC. Ωστόσο, λειτουργεί και με τάση 5 Volt DC. Η κατανάλωση ρεύματος όπως προαναφέρθηκε, είναι στα 4 mA. Πρόκειται για αισθητήρα αναλογικού τύπου με διαστάσεις 1.58 cm διάμετρο και 0.31 cm πάχος . Αναλυτικότερα απεικονίζονται σε μετέπειτα κεφάλαιο οι ενδεικτικές τιμές των καρδιακών παλμών για διάφορα επίπεδα σωματικής δραστηριότητας. Η αρχή λειτουργίας του συγκεκριμένου αισθητήρα παρουσιάζεται στην παρακάτω εικόνα, η οποία στη συνέχεια θα αναλυθεί.[52]

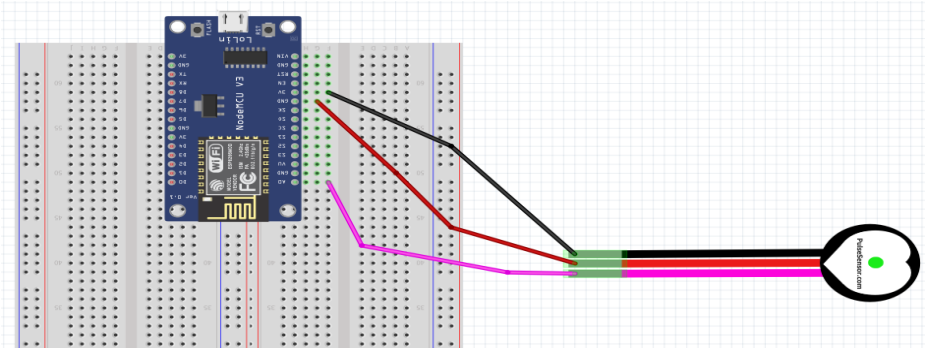


Εικόνα 46: Αρχή λειτουργίας αισθητήρα καρδιακών παλμών

Πηγή : Αρχή λειτουργίας αισθητήρα καρδιακών παλμών

Η διαδικασία που περιγράφεται φέρει την ονομασία ‘Φωτοπλασματογραφία’. Ουσιαστικά, ο αισθητήρας εκπέμπει ένα κόκκινο ή υπέρυθρο φως για να φωτίσει το δέρμα και έναν φωτοανιχνευτή για να εντοπίσει τις διακυμάνσεις στην ένταση του φωτός στη συγκεκριμένη περιοχή. Το φως που χρησιμοποιείται για τη διαδικασία <<ταξιδεύει>> μέσω των ιστών και απορροφάται από χρωστικές ουσίες, οστά και αίμα. Ο αισθητήρας αυτός παρατηρεί οπτικά τη μεταβολή του όγκου της ροής του αίματος, ανιχνεύοντας τις αλλαγές στην ένταση του φωτός. Κάθε καρδιακός παλμός εμφανίζεται ως μία κορυφή , όπως φαίνεται και στο παραπάνω σχήμα.[53]

Παρακάτω παρουσιάζεται η διασύνδεση του αισθητήρα καρδιακών παλμών με το NodeMCU με το πρόγραμμα Fritzing.



Εικόνα 47 : Διασύνδεση αισθητήρα καρδιακών παλμών με NodeMCU

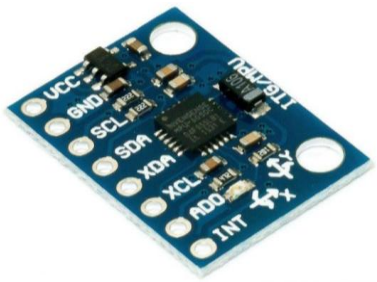
Όπως διακρίνεται και από το παραπάνω σχηματικό του προγράμματος Fritzing, από το μαύρο καλώδιο του αισθητήρα (VCC) συνδέεται στο 3Volt του NodeMCU, από το κόκκινο καλώδιο του αισθητήρα (GND) στο Ground (GND) της πλακέτας. Τέλος, από το ροζ καλώδιο του αισθητήρα (Signal), συνδέεται στο A0 της πλακέτας.

3.2.4 Αισθητήρας μέτρησης βημάτων

Ο αισθητήρας που χρησιμοποιήθηκε για τη μέτρηση των βημάτων του χρήστη, απεικονίζεται στην παρακάτω εικόνα και φέρει την ονομασία MPU-6050. Διαθέτει 8 pin (VCC, GND, SCL, SDA, XDA, XCL, AD0, INT). Το Mpu-6050 είναι ένα τρι-άξονα +/- 2g, +/- 4g, +/-8g, +/-16g. Αυτή η μονάδα περιλαμβάνει ένα γυροσκόπιο 3 αξόνων υψηλής ακρίβειας και ένα επιταχυνσιόμετρο 3 αξόνων. Ο αισθητήρας έχει διαστάσεις 25.5 x 15.2 x 2.48 mm , τάση λειτουργίας από 2.3 - 3.4 Volt DC και βάρος 2.1g. Χρησιμοποιεί ψηφιακό πρωτόκολλο επικοινωνίας με το chip (I2C) και διαθέτει εσωτερικό ρυθμιστή χαμηλής αποκοπής. Λειτουργεί σε συνθήκες θερμοκρασίας από -40 °C έως 85 °C. Η κατανάλωση

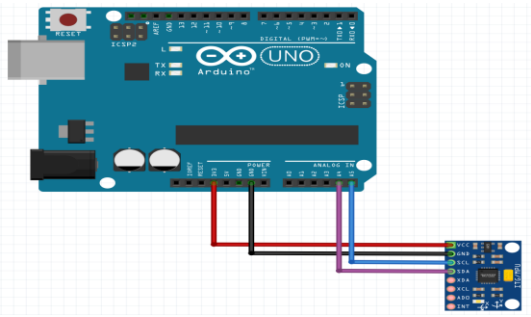
ενέργειάς του σε κατάσταση αναμονής είναι 5μA και σε κατάσταση λειτουργίας είναι 3.6 mA.[54]

Στην παρακάτω εικόνα παρουσιάζεται ο αισθητήρας βημάτων MPU-6050, η διασύνδεσή του με το Arduino Uno, οι βιβλιοθήκες που χρειάστηκαν καθώς και ο κώδικάς του.



Εικόνα 48: Αισθητήρας μέτρησης βημάτων

Πηγή : Αισθητήρας μέτρησης βημάτων



Εικόνα 49 : Διασύνδεση αισθητήρα MPU 6050 με Arduino Uno

Το κόκκινο καλώδιο του αισθητήρα συνδέεται από το άκρο VCC στο 3 Volt της πλακέτας, το μαύρο από το Ground (GND) του αισθητήρα στο Ground του Arduino Uno. Το μπλε καλώδιο του αισθητήρα SCL συνδέεται στην αναλογική είσοδο της πλακέτας A5. Τέλος, το μωβ καλώδιο SDA του αισθητήρα, συνδέεται στην αναλογική είσοδο του αισθητήρα A4. Για τη λειτουργία του αισθητήρα, χρησιμοποιήθηκαν οι βιβλιοθήκες “Wire.h” και η “chris_pedometer.h” .

3.3 Κόστος κατασκευής ‘ SmartWristband ’

Παρακάτω παρουσιάζεται η λίστα με τα υλικά που χρησιμοποιήθηκαν καθώς και το κόστος αυτών.

- ➢ Αισθητήρας θερμοκρασίας-υγρασίας → 11 €
- ➢ Αισθητήρας καρδιακών παλμών → 7.45 €
- ➢ Arduino(Geekcreit) Uno Rev 3 Board → 13.04 €
- ➢ NodeMCU → 9.9 €
- ➢ Αισθητήρας μέτρησης βημάτων → 4.9 €
- ➢ 2 x Καλώδιο σύνδεσης με Η/Υ → 3 €
- ➢ 2 x Logilink Usb Repeater → 20 €

Το συνολικό κόστος της κατασκευής ανήλθε στα 69.3 ευρώ.

ΚΕΦΑΛΑΙΟ 4ο

4.1 Σενάριο καταγραφής θερμοκρασίας - υγρασίας, καρδιακών παλμών και μεταφορά αυτών σε ThingSpeak - IoT

Το κύκλωμα αφορά το NodeMCU, το οποίο λαμβάνει μετρήσεις από τον αισθητήρα θερμοκρασίας - υγρασίας και των καρδιακών παλμών. Τα δεδομένα των παραπάνω αισθητήρων ανανεώνονται κάθε 1 λεπτό, διότι αυτή είναι η χρονική δυνατότητα που παρέχεται δωρεάν στην πλατφόρμα ThingSpeak - IoT. Σε περίπτωση που θέλουμε η ανανέωση των δεδομένων να γίνεται σε πιο σύντομο χρονικό διάστημα ήτοι 3 sec, η ετήσια χρέωση ανέρχεται περίπου στα 220 ευρώ, όπως φαίνεται και στον παρακάτω σύνδεσμο : https://thingspeak.com/prices/thingspeak_student . Ακολουθεί ο κώδικας υλοποίησης της συγκεκριμένης εφαρμογής καθώς και η απεικόνισή του με το πρόγραμμα Fritzing.

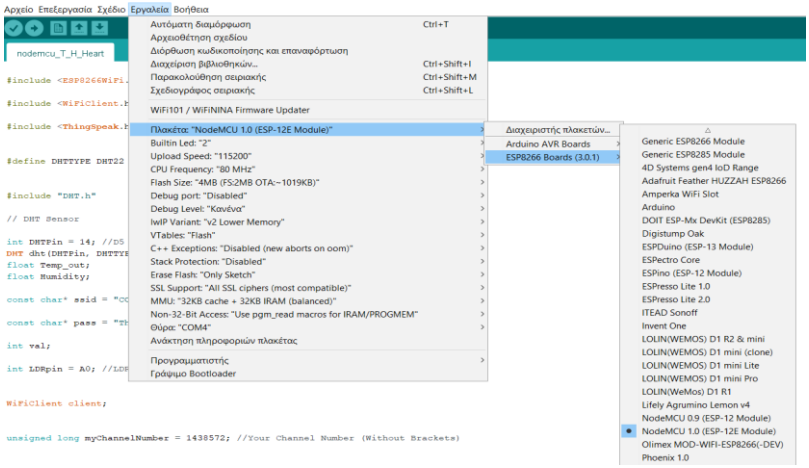
Τα υλικά που χρησιμοποιήθηκαν για την παρακάτω υλοποίηση είναι τα εξής :

- NodeMCU
- Breadboard
- Αισθητήρας θερμοκρασίας - υγρασίας DHT22
- Αισθητήρας καρδιακών παλμών

Οι βιβλιοθήκες που χρειάζονται για τη λειτουργία του παραπάνω συστήματος είναι η DHT.h , η ESP8266WiFi.h , η WiFiClient.h και η ThingSpeak.h .

Το βασικότερο στοιχείο το οποίο δεν πραγματοποιήθηκε λόγω του υψηλού κόστους του είναι η χρήση μικρότερων αισθητήρων, μπαταριών και πλακέτας arduino nodemcu ,ώστε το SmartWristband να αποκτήσει φορητότητα.

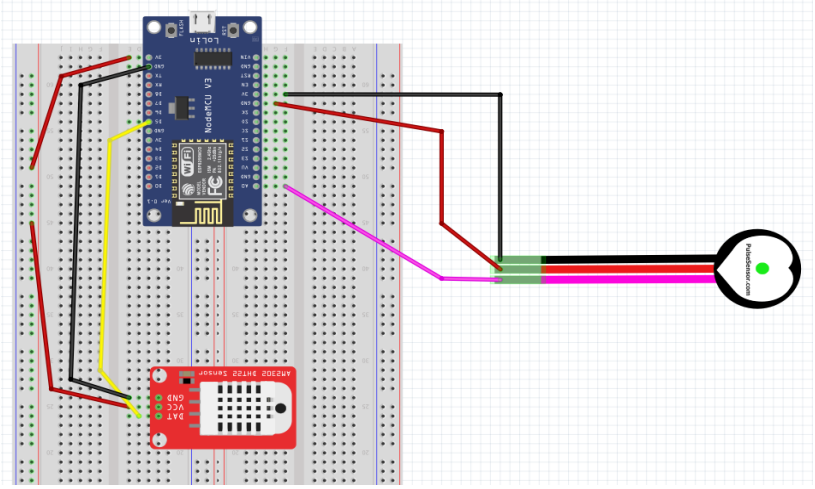
Για την επιλογή της πλακέτας του κυκλώματος ακολουθούμε τα εξής βήματα από το περιβάλλον ανάπτυξης του Arduino IDE όπως φαίνονται παρακάτω.



Εικόνα 50 : Επιλογή πλακέτας Node MCU

Πηγή : Πρόγραμμα Arduino IDE

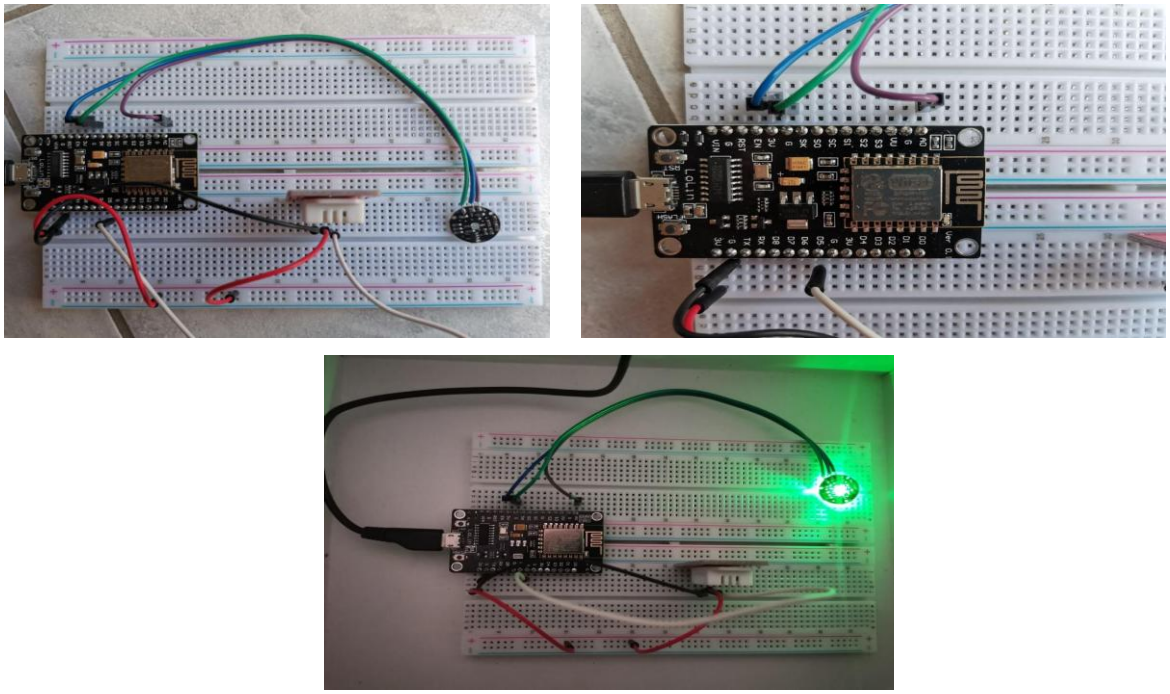
Με τη βοήθεια του προγράμματος Fritzing κατασκευάστηκε το παραπάνω κύκλωμα σε ψηφιακή μορφή , το οποίο παρουσιάζεται στην επόμενη εικόνα.



Εικόνα 51: Κύκλωμα μέτρησης θερμοκρασίας - υγρασίας και καρδιακών παλμών

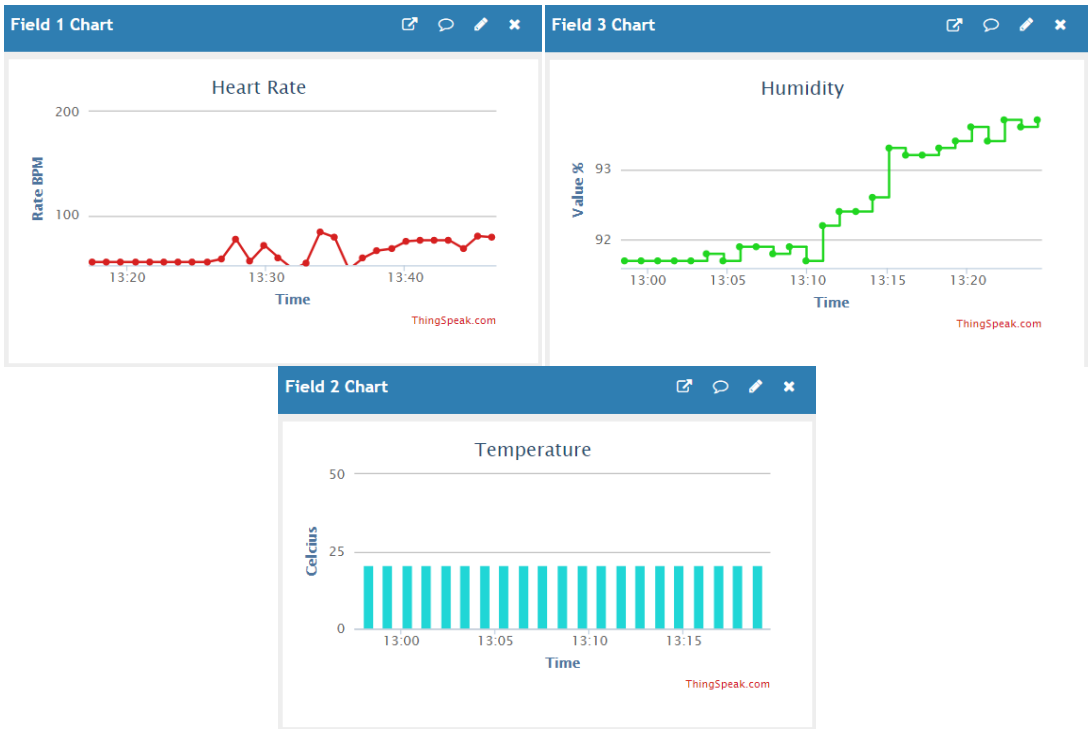
Το κίτρινο καλώδιο του αισθητήρα DHT22 συνδέεται στο D5 του NodeMCU, το μαύρο καλώδιο GND στο άκρο GND της πλακέτας και το κόκκινο καλώδιο VCC συνδέεται στο άκρο των 3 Volt του NodeMCU. Το ροζ καλώδιο του αισθητήρα των καρδιακών παλμών συνδέεται στο A0 της πλακέτας, το κόκκινο καλώδιο Ground (GND) στο GND της πλακέτας και το μαύρο καλώδιο VCC στο άκρο των 3 Volt του NodeMCU.

Το κύκλωμα που κατασκευάστηκε για τη μέτρηση της θερμοκρασίας - υγρασίας και των καρδιακών παλμών του χρήστη, απεικονίζεται παρακάτω.



Εικόνα 52 : Κύκλωμα NodeMCU, DHT22, Heart Rate

Παρακάτω, παρουσιάζεται ένα δείγμα μετρήσεων του συγκεκριμένου κυκλώματος όπως αυτό προβάλλεται στο ThingSpeak IoT καθώς και τα διαγράμματα gauge αυτών.



Εικόνα 53: Μετρήσεις καρδιακών παλμών, θερμοκρασίας – υγρασίας

Πηγή : *Μετρήσεις καρδιακών παλμών, θερμοκρασίας -υγρασίας*

Εικόνα 54: Widget Gauge θερμοκρασίας, υγρασίας, καρδιακών παλμών

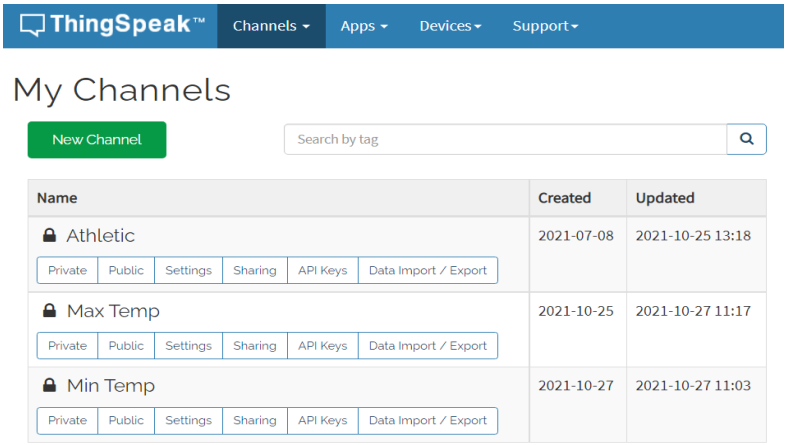
Πηγή : Widget Gauge θερμοκρασίας, υγρασίας, καρδιακών παλμών

4.1.1 Εύρεση Μέγιστης – Ελάχιστης θερμοκρασίας και Μέσης τιμής υγρασίας και συνδυασμοί διαγραμμάτων

ΜΕΓΙΣΤΗ ΤΙΜΗ ΘΕΡΜΟΚΡΑΣΙΑΣ

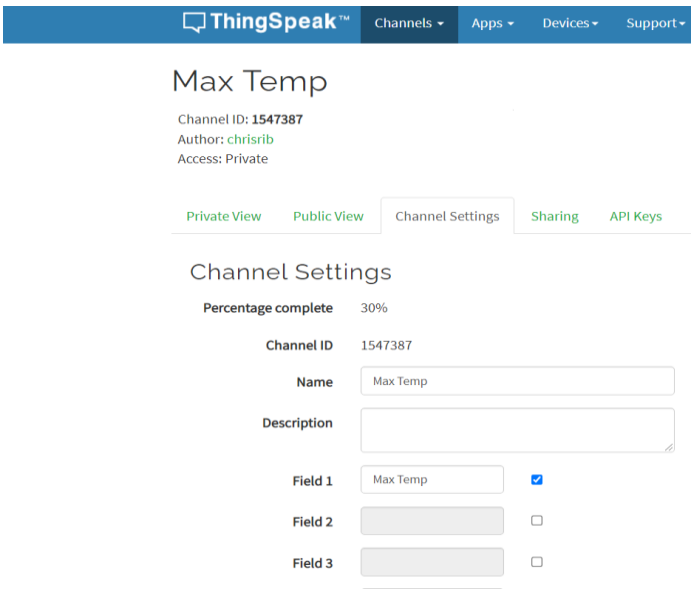
Για την εύρεση της μέγιστης θερμοκρασίας χρησιμοποιήθηκε η δυνατότητα του ThingSpeak IoT με ονομασία Matlab Analysis, η οποία αναλύει τα δεδομένα ενός καναλιού που θα ορίσει ο προγραμματιστής. Κάθε κανάλι ξεχωρίζει από τα υπόλοιπα χάρη στην "ταυτότητα" που το διακατέχει, το λεγόμενο Channel ID. Τα δεδομένα της θερμοκρασίας στη δική μας περίπτωση στέλνονται από το NodeMCU στο δεύτερο διάγραμμα - Chart του πρώτου καναλιού με **Channel ID : 1438572** και **Field ID : 2** . Η τιμή της μέγιστης θερμοκρασίας αποθηκεύεται σε άλλο κανάλι με **Channel ID : 1547387** . Για τη δημιουργία του καναλιού ακολουθήθηκαν τα παρακάτω βήματα.

Όπως φαίνεται παρακάτω από την καρτέλα My Channels του ThingSpeak IoT, επιλέγουμε τη δημιουργία Νέου Καναλιού - New Channel.



Εικόνα 55 : Δημιουργία νέου καναλιού
Πηγή : Δημιουργία νέου καναλιού ThingSpeak

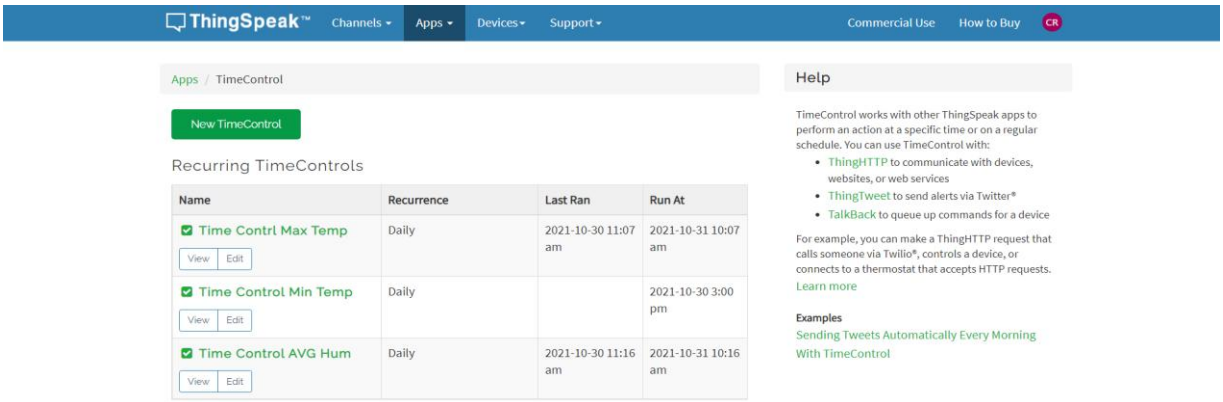
Στη συνέχεια, προσθέτουμε όλες τις απαραίτητες πληροφορίες σχετικά με το κανάλι όπως η ονομασία, η περιγραφή του καναλιού κλπ. Ακόμη σε αυτό το σημείο μπορούμε να ορίσουμε την ιδιωτικότητα του καναλιού, δηλαδή αν τα δεδομένα θα προβάλλονται μόνο στο χρήστη ή θα είναι δημόσια ,από την καρτέλα ρυθμίσεων Sharing. Επιπλέον, στην καρτέλα API Keys παρουσιάζονται τα κλειδιά που χρειάζονται στον κώδικα για την προσθήκη των τιμών στο κανάλι. Τέλος, αποθηκεύουμε τις ρυθμίσεις με την επιλογή Save Channel στο τέλος της σελίδας.



Εικόνα 56 : Ρυθμίσεις καναλιού ThingSpeak IoT
Πηγή : Ρυθμίσεις καναλιού ThingSpeak IoT

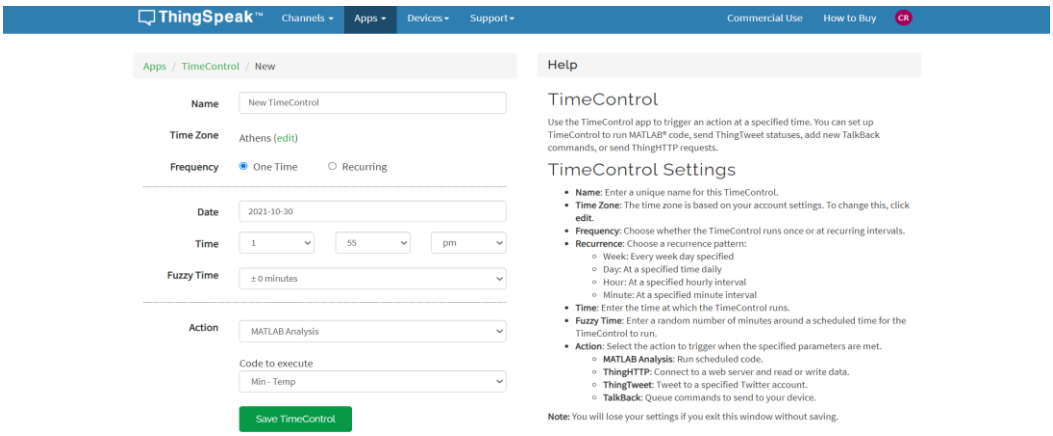
Προκειμένου ο κώδικας της εύρεσης της μέγιστης θερμοκρασίας να "τρέχει" σε καθημερινή βάση, χρησιμοποιήθηκε η δυνατότητα του ThingSpeak IoT με ονομασία Time

Control. Αυτή η λειτουργία επιτρέπει στον προγραμματιστή να ορίσει τη συχνότητα στην οποία θα "τρέχει" ο κώδικας της κάθε εφαρμογής, δηλαδή μόνο μία φορά, σε καθημερινή βάση, σε εβδομαδιαία βάση ή σε καθορισμένη μέρα και ώρα. Για τη δημιουργία του Time Control ακολουθήθηκαν τα παρακάτω βήματα. Από την καρτέλα Apps επιλέγουμε την επιλογή Time Control και μας εμφανίζεται η επόμενη σελίδα.



Εικόνα 57 : Μενού Time Control
Πηγή : Μενού Time Control

Στη συνέχεια επιλέγουμε το New Time Control και μας εμφανίζεται το παρακάτω παράθυρο ρυθμίσεων.



Εικόνα 58 : Παράθυρο ρυθμίσεων Time Control
Πηγή : Παράθυρο ρυθμίσεων Time Control

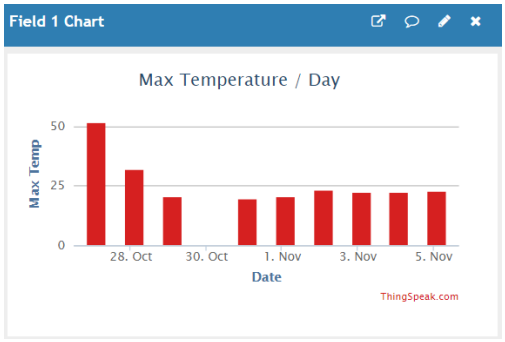
Από αυτό το παράθυρο μπορούμε να ορίσουμε την ονομασία του Time Control, τη συχνότητα αυτής της ενέργειας καθώς και το πού θα αναφέρεται αυτή η ενέργεια. Στη δική μας περίπτωση η ενέργεια αυτού του Time Control αφορά τον κώδικα στο Matlab Analysis με ονομασία Max Temp για την εύρεση της μέγιστης θερμοκρασίας και έχει οριστεί να εμφανίζει τη μέγιστη τιμή κάθε βράδυ. Ολοκληρώνοντας όλες τις απαραίτητες ρυθμίσεις

επιλέγουμε Save Time Control για να τις κατοχυρώσουμε. Με τον ίδιο ακριβώς τρόπο δημιουργήθηκε Time Control για την εύρεση της ελάχιστης τιμής της θερμοκρασίας καθώς και της μέσης τιμής της υγρασίας που ακολουθεί παρακάτω.

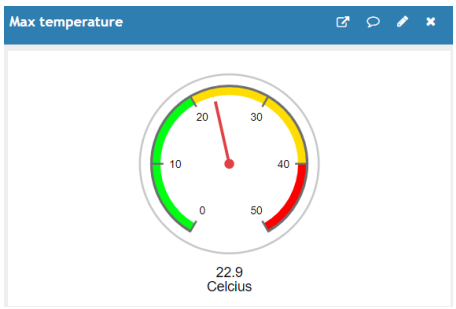
Ο κώδικας που χρησιμοποιήθηκε για την εύρεση της μέγιστης θερμοκρασίας στο Matlab Analysis παρουσιάζεται παρακάτω.

```
readChannelID = 1438572; %Προσθήκη του channel ID απ' όπου θα πάρει δεδομένα

TemperatureFieldID = 2; %Πρσθήκη του Field ID που αντιστοιχεί στη θερμοκρασία

readAPIKey = 'AF52JRB928DGQ5BS'; %Προσθήκη του readAPIKey του καναλιού που θα πάρει δεδομένα


[tempF,timeStamp] =
thingSpeakRead(readChannelID,'Fields',TemperatureFieldID, ...

'numDays',1,'ReadKey',readAPIKey); %Διαβάζει δεδομένα θερμοκρασίας μίας ημέρας

[maxTempF,maxTempIndex] = max(tempF); %Μέγιστη θερμοκρασία

timeMaxTemp = timeStamp(maxTempIndex); %Χρονική στιγμή μέγιστης θερμοκρασίας

display(maxTempF,'Maximum Temperature for the past 24 hours is');

writeChannelID = 1547387; %Προσθήκη του WriteChannelID στο οποίο θα γραφτούν τα δεδομένα

writeAPIKey = 'PQHMCFD8CE4UEKP1'; %Πρσθήκη του WriteAPIKey του καναλιού που θα γραφτουν τα δεδομένα


thingSpeakWrite(writeChannelID,maxTempF,'timestamp',timeMaxTemp,'WriteKey',
writeAPIKey); %Στέλνει τα δεδομένα στο κανάλι που έχει οριστεί παραπάνω
```

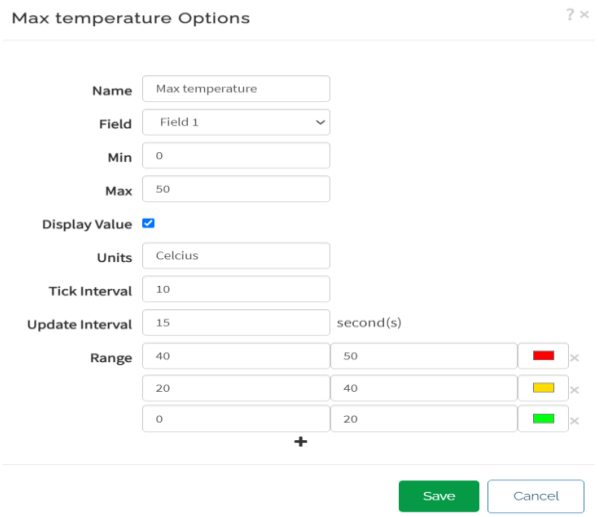
Επίσης, για την καλύτερη απεικόνιση της μέγιστης θερμοκρασίας δημιουργήθηκαν 2 Widgets. Συγκεκριμένα, δημιουργήθηκε ένα διάγραμμα Gauge με χρώματα ανάλογα της μέγιστης θερμοκρασίας και ένα Lamp Indicator, το οποίο ανάβει πράσινο χρώμα σε περίπτωση που η μέγιστη θερμοκρασία ξεπεράσει τους 20 βαθμούς Κελσίου. Οι ρυθμίσεις των Gauge και Lamp Indicator παρουσιάζονται παρακάτω.



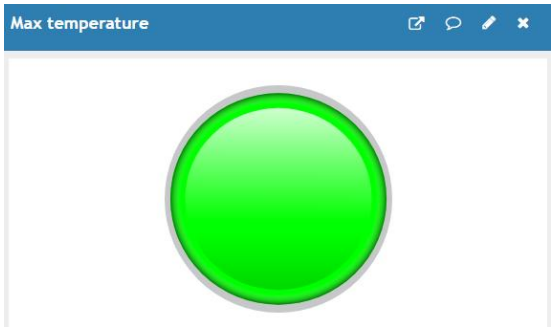
Εικόνα 59 : Μέγιστη θερμοκρασία ανά ημέρα
Πηγή : [Μέγιστη θερμοκρασία ανά ημέρα]



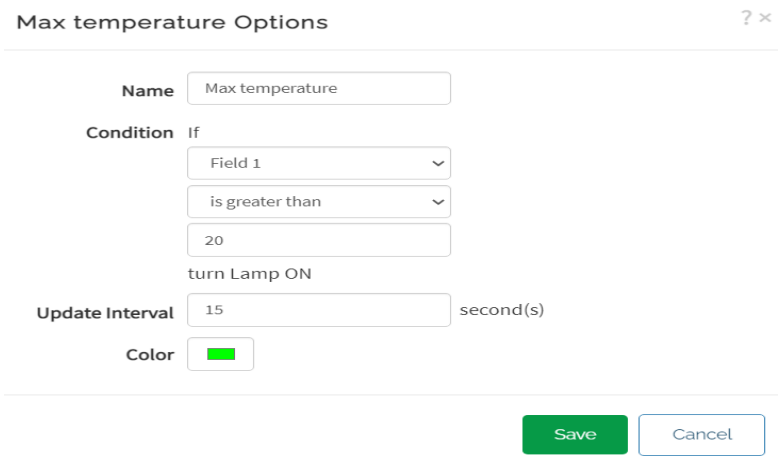
Εικόνα 60 : Gauge Widget μέγιστης θερμοκρασίας
Πηγή : [Gauge Widget μέγιστης θερμοκρασίας]



Εικόνα 61 : Ρυθμίσεις Gauge μέγιστης θερμοκρασίας
Πηγή : [Ρυθμίσεις Gauge]



Εικόνα 62 : Lamp Indicator μέγιστης θερμοκρασίας
Πηγή : Lamp Indicator μέγιστης θερμοκρασίας



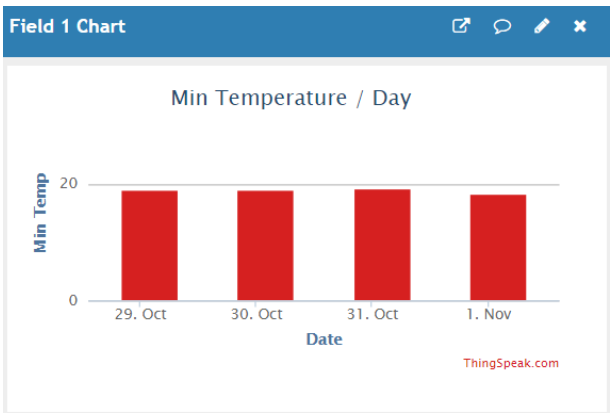
Εικόνα 63 : Ρυθμίσεις Lamp Indicator μέγιστης θερμοκρασίας
Πηγή : Ρυθμίσεις Lamp Indicator

ΕΛΑΧΙΣΤΗ ΤΙΜΗ ΘΕΡΜΟΚΡΑΣΙΑΣ

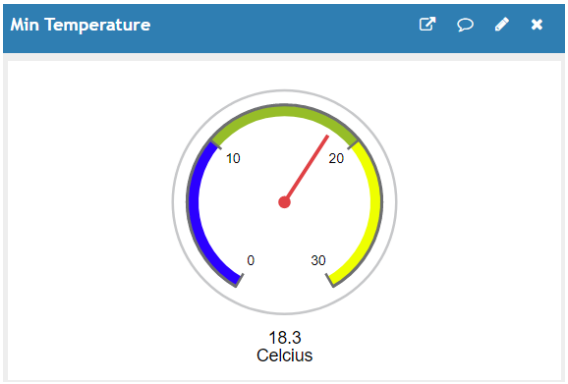
Για την εύρεση της ελάχιστης θερμοκρασίας ανά ημέρα χρησιμοποιήθηκε ξανά η δυνατότητα του ThingSpeak IoT, το Matlab Analysis. Σε αυτή την περίπτωση τα δεδομένα διαβάζονται από το βασικό κανάλι με **Channel ID : 1438572** και συγκεκριμένα από το δεύτερο διάγραμμα **Field : 2** το οποίο αντιστοιχεί στο διάγραμμα της θερμοκρασίας. Η τιμή της ελάχιστης θερμοκρασίας αποθηκεύεται στο κανάλι με **Channel ID : 1549959**. Η διαδικασία που ακολουθήθηκε για τη δημιουργία του καναλιού είναι η ίδια με αυτή της μέγιστης τιμής της θερμοκρασίας. Στο κανάλι της ελάχιστης θερμοκρασίας τοποθετήθηκαν για την καλύτερη προβολή της τιμής ένα διάγραμμα Gauge με χρώματα ανάλογα με την τιμή της ελάχιστης θερμοκρασίας και ένα Lamp Indicator το οποίο σε περίπτωση που η θερμοκρασία πέσει κάτω από τους 10 βαθμούς Κελσίου ανάβει μπλε χρώμα. Οι ρυθμίσεις αυτών καθώς και ο κώδικας παρουσιάζονται παρακάτω.

Ο κώδικας που χρησιμοποιήθηκε για την εύρεση της ελάχιστης θερμοκρασίας παρουσιάζεται παρακάτω.

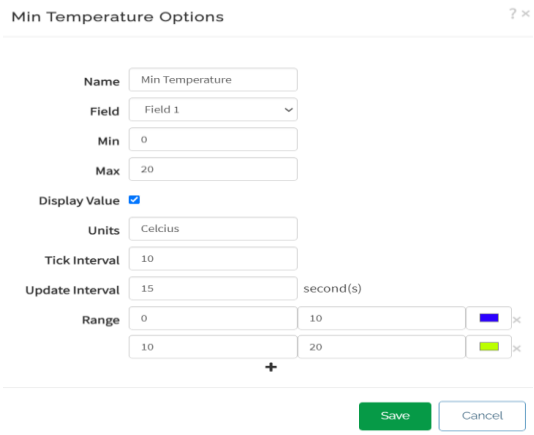
```
readChannelID = 1438572; %Προσθήκη του channel ID απ' όπου θα πάρει δεδομένα

TemperatureFieldID = 2; %Πρσθήκη του Filed ID που αντιστοιχεί στη θερμοκρασία


readAPIKey = 'AF52JRB928DGQ5BS'; %Προσθήκη του readAPIKey του καναλιού που θα πάρει δεδομένα


[tempF,timeStamp] =
thingSpeakRead(readChannelID,'Fields',TemperatureFieldID, ...

'numDays',1,'ReadKey',readAPIKey); %Διαβάζει δεδομένα θερμοκρασίας μίας ημέρας

[minTempF,minTempIndex] = min(tempF); %Ελάχιστη θερμοκρασία

timeMinTemp = timeStamp(minTempIndex); %Χρονική στιγμή ελάχιστης θερμοκρασίας

display(minTempF,'Minimum Temperature for the past 24 hours is');

writeChannelID = 1549959; %Προσθήκη του WriteChannelID στο οποίο θα γραφτούν τα δεδομένα

writeAPIKey = 'S2NX5FNRAP7T48UY'; %Πρσθήκη του WriteAPIKey του καναλιού που θα γραφτούν τα δεδομένα

thingSpeakWrite(writeChannelID,minTempF,'timestamp',timeMinTemp,'WriteKey',
writeAPIKey); %Στέλνει τα δεδομένα στο κανάλι που έχει οριστεί παραπάνω
```



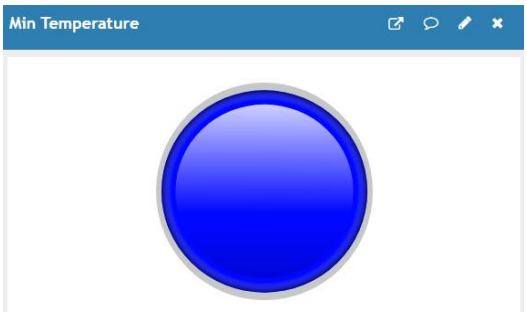
Εικόνα 64 : Ελάχιστη θερμοκρασία ανά ημέρα
Πηγή : Ελάχιστη θερμοκρασία ανά ημέρα



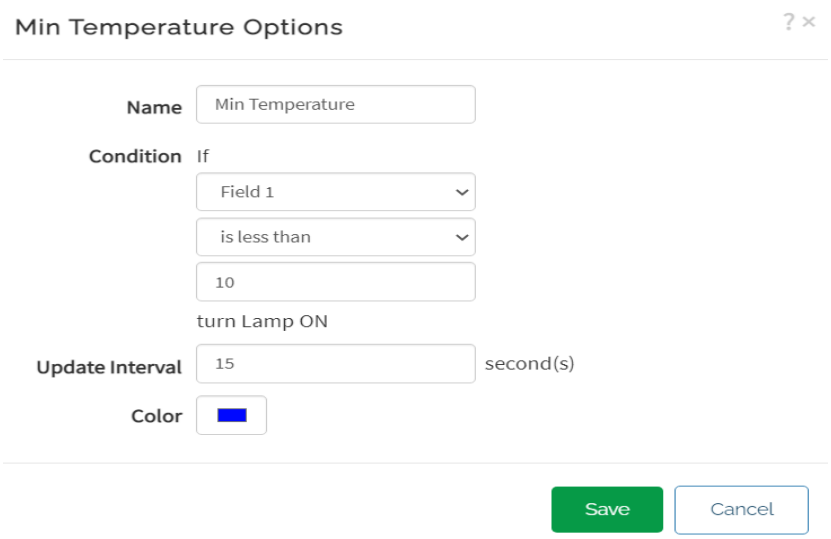
Εικόνα 65 : Gauge Widget ελάχιστης θερμοκρασίας
Πηγή : Gauge Widget ελάχιστης θερμοκρασίας



Εικόνα 66 : Ρυθμίσεις Gauge ελάχιστης θερμοκρασίας
Πηγή : Ρυθμίσεις Gauge



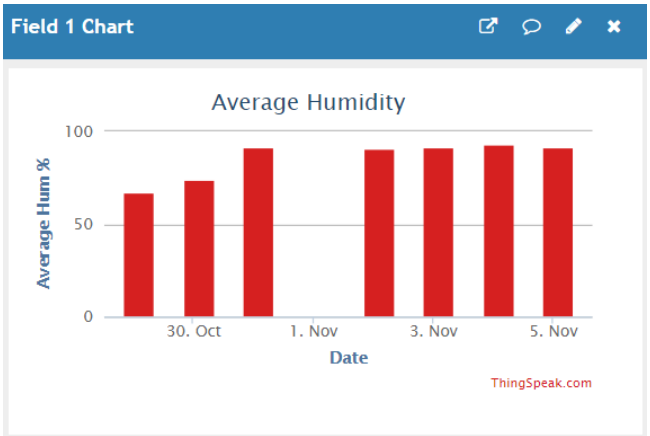
Εικόνα 67 : Lamp Indicator ελάχιστης θερμοκρασίας
Πηγή : Lamp Indicator ελάχιστης θερμοκρασίας



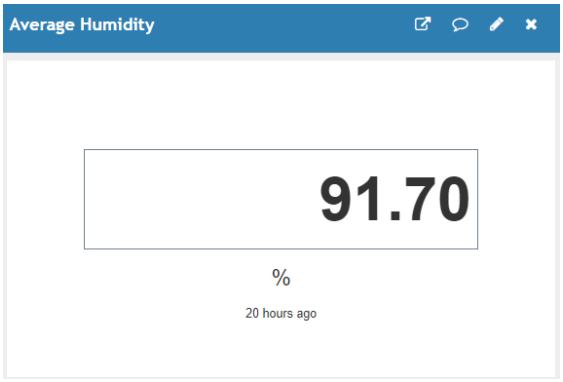
Εικόνα 68 : Ρυθμίσεις Lamp Indicator ελάχιστης τιμής
Πηγή : Ρυθμίσεις Lamp Indicator

ΜΕΣΗ ΤΙΜΗ ΥΓΡΑΣΙΑΣ

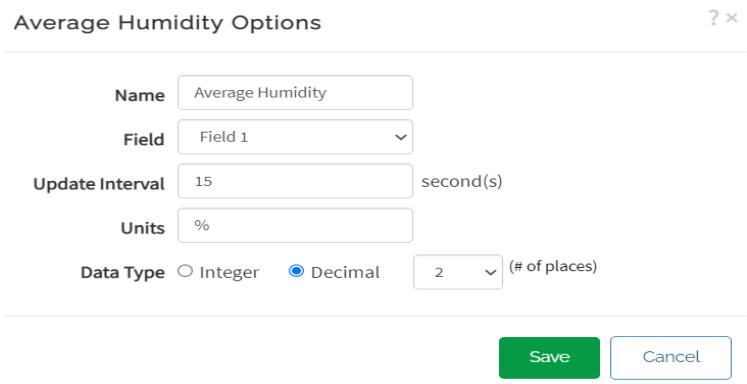
Για την εύρεση της μέσης τιμής της υγρασίας χρησιμοποιήθηκε η επιλογή του ThingSpeak IoT , το Matlab Analysis. Τα δεδομένα για αυτή την ανάλυση στέλνονται από το NodeMCU στο κανάλι με **Channel ID : 1438572** και συγκεκριμένα στο διάγραμμα **Field ID 2**. Η τιμή της μέσης υγρασίας θα αποθηκεύεται σε άλλο κανάλι με **Channel ID : 1552376**. Για τη δημιουργία του καναλιού ακολουθήθηκαν τα ίδια βήματα με τις δύο προηγούμενες αναλύσεις. Στο κανάλι της μέσης τιμής της υγρασίας τοποθετήθηκε για την καλύτερη απεικόνισή της ένα Numeric Widget. Η ρύθμιση αυτού και ο κώδικας παρουσιάζονται παρακάτω.



Εικόνα 69 : Μέση υγρασία ανά ημέρα
Πηγή : Μέση υγρασία ανά ημέρα



Εικόνα 70 : Numeric Widget μέσης υγρασίας
Πηγή : Numeric Widget μέσης υγρασίας



Εικόνα 71 : Ρυθμίσεις του Numeric Widget
Πηγή : Ρύθμιση Numeric Widget

Ο κώδικας που χρησιμοποιήθηκε για την εύρεση της μέσης τιμής της υγρασίας απεικονίζεται παρακάτω.

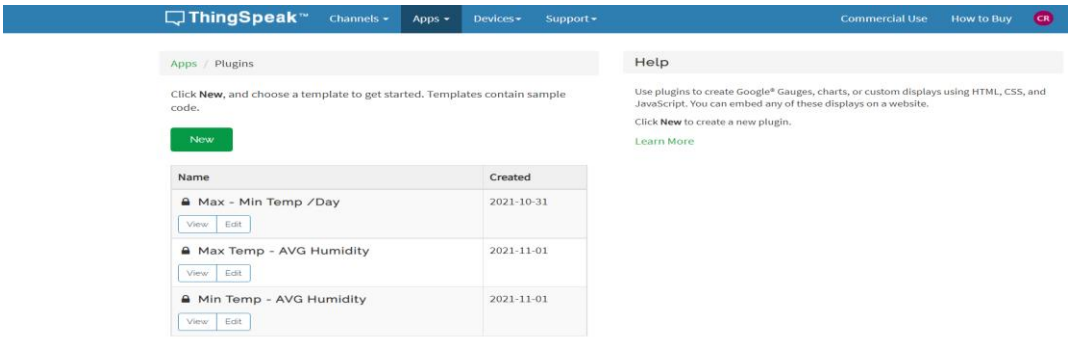
```
readChannelID = 1438572; %Προσθήκη του Channel ID  απ 'όπου θα διαβαζει δεδομένα

humidityFieldID = 3; %Προσθήκη του Field ID απ' όπου θα διαβαζει δεδομένα

readAPIKey = 'AF52JRB928DGQ5BS'; %Προσθήκη του ReadAPI Key για να διαβάσει δεδομένα απ΄το κανάλι

humidity = thingSpeakRead(readChannelID,'Fields',humidityFieldID,'NumMinutes',60,'ReadKey',readAPIKey); %Διαβάζει τα δεδομένα της υγρασίας


avgHumidity = mean(humidity, 'omitnan'); %Υπολογίζει τη μέση τιμή της υγρασία
display(avgHumidity,'Average Humidity');


writeChannelID = 1552376; %Προσθήκη του Channel ID για να στείλει τα δεδομένα
```

```
writeAPIKey = 'QFSMX8XQSERIG9WC'; %Προσθήκη του WriteAPI Key του καναλιού για να γράψει τα δεδομένα


thingSpeakWrite(writeChannelID,avgHumidity,'writeKey',writeAPIKey);
%Στέλνει τα δεδομένα στο κανάλι που ορίστηκε
```

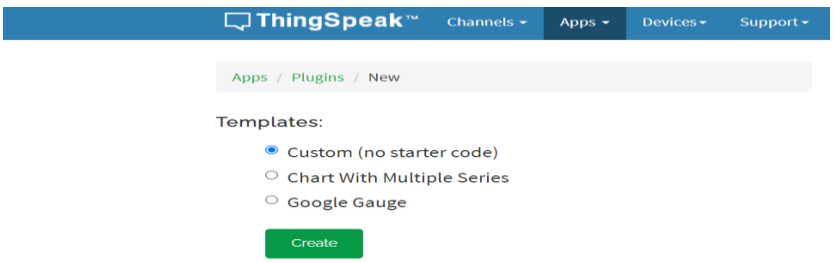
ΣΥΝΔΥΑΣΜΟΙ ΔΙΑΓΡΑΜΜΑΤΩΝ

1. Μέγιστη – ελάχιστη θερμοκρασία ανά ημέρα

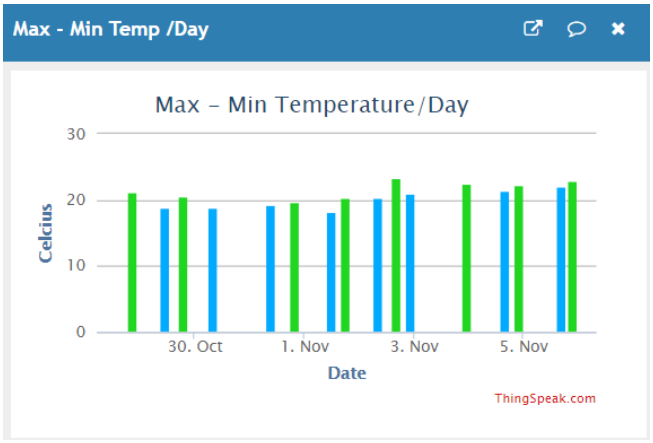
Για τη δημιουργία κοινού διαγράμματος για την απεικόνιση της μέγιστης - ελάχιστης θερμοκρασίας ανά ημέρα, χρησιμοποιήθηκε η δυνατότητα του ThingSpeak IoT με ονομασία Plugins. Με τη δυνατότητα αυτή μπορούμε να αναπαραστήσουμε σε κοινό διάγραμμα δύο ή περισσότερα γραφήματα. Από την κεντρική καρτέλα του ThingSpeak IoT επιλέγουμε το Apps και στη συνέχεια το Plugins. Έπειτα, επιλέγουμε το New και τη διαθέσιμη επιλογή Chart With Multiple Series. Τα βήματα που ακολουθήθηκαν παρουσιάζονται παρακάτω.



Εικόνα 72 : Δημιουργία γραφήματος
Πηγή : Δημιουργία γραφήματος Plugin



Εικόνα 73 : Επιλογή του Plugin
Πηγή : Επιλογή του Plugin



Εικόνα 74 : Μέγιστη – Ελάχιστη θερμοκρασία ανά ημέρα
Πηγή : Μέγιστη – Ελάχιστη θερμοκρασία ανά ημέρα

Ο κώδικας που χρησιμοποιήθηκε για την αναπαράσταση της μέγιστης - ελάχιστης θερμοκρασίας ανά ημέρα σε κοινο διάγραμμα παρουσιάζεται παρακάτω.

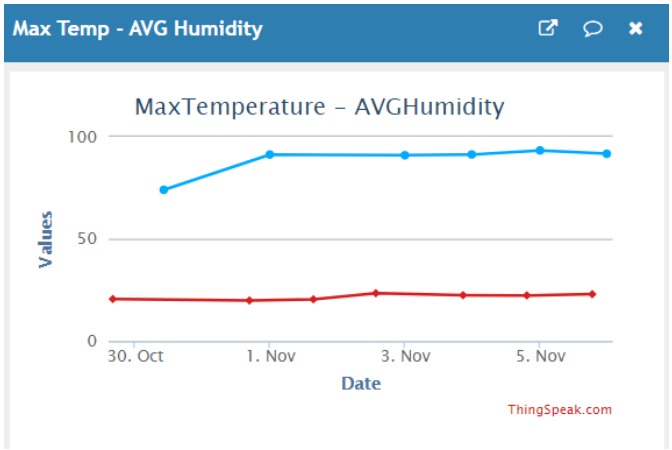
```
<script type="text/javascript">

  var series_1_channel_id = 1547387; //Στοιχεία καναλιού μέγιστης θερμοκρασίας
  var series_1_field_number = 1;
  var series_1_read_api_key = '1EK70LCCC8RQCAXI';
  var series_1_results = 7;
  var series_1_color = '#20d620';

  var series_2_channel_id = 1549959; //Στοιχεία καναλιού ελάχιστης θερμοκρασίας
  var series_2_field_number = 1;
  var series_2_read_api_key = 'IKQ7UFLWJQRYJ8S6';
  var series_2_results = 7;
  var series_2_color = '#00aaff';

  var chart_title = 'Max - Min Temperature/Day'; //Τίτλος του διαγράμματος
  var y_axis_title = 'Celcius'; //Τίτλος στον άξονα y

  var my_offset = new Date().getTimezoneOffset(); //Διαβάζει την ώρα του χρήστη
  // chart variable
  var my_chart;

  // when the document is ready
  $(document).on('ready', function() {
    // add a blank chart
    addChart();
    addSeries(series_1_channel_id, series_1_field_number, series_1_read_api_key, series_1_results, series_1_color); //Προσθέτει στην πρώτη σειρά τα δεδομένα του πρώτου καναλιού
    addSeries(series_2_channel_id, series_2_field_number, series_2_read_api_key, series_2_results, series_2_color); //Προσθέτει στη δεύτερη σειρά τα δεδομένα του δεύτερου καναλιού
  });
```

```
  // add the base chart
  function addChart() {
    var localDate;

    var chartOptions = { //Χαρακτηριστικά για το γράφημα
      chart: {
        renderTo: 'chart-container',
        defaultSeriesType: 'column',
        backgroundColor: '#ffffff',
        events: { }
      },
      title: { text: chart_title },
      plotOptions: {
        series: {
          marker: { radius: 3 },
          animation: true,
          step: false,
          borderWidth: 0,
          turboThreshold: 0
        }
      },
      tooltip: {
        formatter: function() { //Υπολογίζει την τοπική ώρα
          var d = new Date(this.x + (my_offset*60000));
          var n = (this.point.name === undefined) ? '' : '<br>' +
this.point.name;
          return this.series.name + ':<b>' + this.y + '</b>' + n + '<br>' +
d.toDateString() + '<br>' + d.toTimeString().replace(/\(.*\)/, "");
        }
      },
      xAxis: { //Δεδομένα στον άξονα x
        type: 'datetime',
        title: { text: 'Date' }
      },
      yAxis: { title: { text: y_axis_title } }, //Δεδομένα στον άξονα y
      exporting: { enabled: false },
      legend: { enabled: false },
      credits: {
        text: 'ThingSpeak.com',
        href: 'https://thingspeak.com/',
        style: { color: '#D62020' }
      }
    };

    my_chart = new Highcharts.Chart(chartOptions); //Δημιουργία του
γραφήματος
  }

  function addSeries(channel_id, field_number, api_key, results, color) {
//Προσθέτει τις γραμμές στο διάγραμμα
    var field_name = 'field' + field_number;

    // get the data with a webservice call
    $.getJSON('https://api.thingspeak.com/channels/' + channel_id +
'/fields/' + field_number + '.json?offset=0&round=2&results=' + results +
'&api_key=' + api_key, function(data) {

      // blank array for holding chart data
      var chart_data = [];
      $.each(data.feeds, function() {
        var point = new Highcharts.Point();
```

```
        var value = this[field_name];
        point.x = getChartDate(this.created_at);
        point.y = parseFloat(value);
        if (this.location) { point.name = this.location; }
        if (!isNaN(parseInt(value))) { chart_data.push(point); }
      });
      my_chart.addSeries({          data:          chart_data,          name:
data.channel[field_name], color: color });
    });
  }
  function getChartDate(d) {
    return Date.parse(d) - (my_offset * 60000);
  }

</script>
```

2. Μέγιστη θερμοκρασία – Μέση υγρασία ανά ημέρα

Με τον ίδιο τρόπο με τον οποίο έγινε η προηγούμενη αναπαράσταση της μέγιστης - ελάχιστης θερμοκρασίας σε κοινό διάγραμμα, δημιουργήθηκε και το γράφημα της απεικόνισης της μέγιστης θερμοκρασίας με τη μέση τιμή της υγρασίας ανά ημέρα. Ο κώδικας για αυτή την περίπτωση παρουσιάζεται παρακάτω.



Εικόνα 75 : Μέγιστη θερμοκρασία – Μέση υγρασία
Πηγή : Μέγιστη θερμοκρασία – Μέση υγρασία

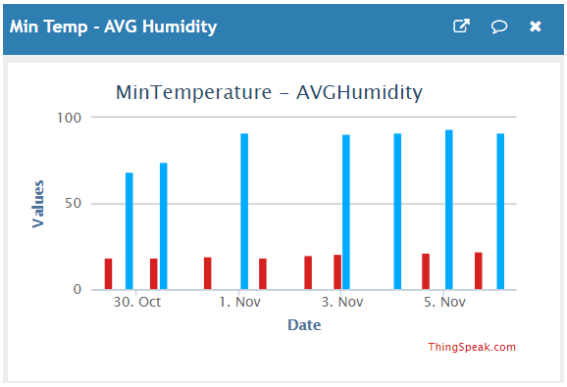
```
<script type="text/javascript">
  var  series_1_channel_id  =  1547387;  //Στοιχεία  καναλιού  μέγιστης
θερμοκρασίας
  var series_1_field_number = 1;
  var series_1_read_api_key = '1EK70LCCC8RQCAXI';
  var series_1_results = 5;
  var series_1_color = '#d62020';

  var series_2_channel_id = 1552376;  //Στοιχεία καναλιού μέσης υγρασίας
  var series_2_field_number = 1;
  var series_2_read_api_key = 'ZTK7QUL41XD2O23M';
```

```
  var series_2_results = 5;
  var series_2_color = '#00aaff';

  var chart_title = 'MaxTemperature - AVGHumidity'; //Τίτλος του διαγράμματος
  var y_axis_title = 'Values'; //Τίτλος στον άξονα y

  var my_offset = new Date().getTimezoneOffset(); //Διαβάζει την ώρα του χρήστη
  // chart variable
  var my_chart;

  // when the document is ready
  $(document).on('ready', function() {
    // add a blank chart
    addChart();
    addSeries(series_1_channel_id, series_1_field_number, series_1_read_api_key, series_1_results, series_1_color); //Προσθέτει στην πρώτη σειρά τα δεδομένα του πρώτου καναλιού
    addSeries(series_2_channel_id, series_2_field_number, series_2_read_api_key, series_2_results, series_2_color); //Προσθέτει στη δεύτερη σειρά τα δεδομένα του δεύτερου καναλιού
  });

  // add the base chart
  function addChart() {
    var localDate;

    var chartOptions = {  //Χαρακτηριστικά για το γράφημα
      chart: {
        renderTo: 'chart-container',
        defaultSeriesType: 'column',
        backgroundColor: '#ffffff',
        events: { }
      },
      title: { text: chart_title },
      plotOptions: {
        series: {
          marker: { radius: 3 },
          animation: true,
          step: false,
          borderWidth: 0,
          turboThreshold: 0
        }
      },
      tooltip: {
        formatter: function() {  //Υπολογίζει την τοπική ώρα
          var d = new Date(this.x + (my_offset*60000));
          var n = (this.point.name === undefined) ? '' : '<br>' + this.point.name;
          return this.series.name + ':<b>' + this.y + '</b>' + n + '<br>' + d.toDateString() + '<br>' + d.toTimeString().replace(/\(.*\)/, "");
        }
      },
      xAxis: {  //Δεδομένα στον άξονα x
        type: 'datetime',
        title: { text: 'Date' }
      },
      yAxis: { title: { text: y_axis_title } },  //Δεδομένα στον άξονα y
      exporting: { enabled: false },
      legend: { enabled: false },
```

```
      credits: {
        text: 'ThingSpeak.com',
        href: 'https://thingspeak.com/',
        style: { color: '#D62020' }
      }
    };
    my_chart = new Highcharts.Chart(chartOptions);     //Δημιουργία του
γραφήματος
  }
  function addSeries(channel_id, field_number, api_key, results, color) {
//Προσθέτει τις γραμμές στο διάγραμμα
    var field_name = 'field' + field_number;

    // get the data with a webservice call
    $.getJSON('https://api.thingspeak.com/channels/' + channel_id +
'/fields/' + field_number + '.json?offset=0&round=2&results=' + results +
'&api_key=' + api_key, function(data) {

      // blank array for holding chart data
      var chart_data = [];
      $.each(data.feeds, function() {
        var point = new Highcharts.Point();
        var value = this[field_name];
        point.x = getChartDate(this.created_at);
        point.y = parseFloat(value);
        if (this.location) { point.name = this.location; }
        if (!isNaN(parseInt(value))) { chart_data.push(point); }
      });
      my_chart.addSeries({          data:          chart_data,          name:
data.channel[field_name], color: color });
    });
  }
  function getChartDate(d) {
    return Date.parse(d) - (my_offset * 60000);
  }

</script>
```

3. Ελάχιστη θερμοκρασία – Μέση υγρασία ανά ημέρα

Όπως ακριβώς και στα δύο προηγούμενα βήματα έτσι και τώρα, για την ταυτόχρονη απεικόνιση της ελάχιστης θερμοκρασίας με τη μέση τιμή της υγρασίας χρειάστηκε η δυνατότητα του ThingSpeak IoT, το Plugins. Ο κώδικας που χρησιμοποιήθηκε σε αυτή την περίπτωση απεικονίζεται παρακάτω.



Εικόνα 76 : Ελάχιστη θερμοκρασία – Μέση υγρασία
Πηγή : Ελάχιστη θερμοκρασία – Μέση υγρασία

```
<script type="text/javascript">
  var series_1_channel_id = 1549959;     //Στοιχεία καναλιού ελάχιστης θερμοκρασίας
  var series_1_field_number = 1;
  var series_1_read_api_key = 'IKQ7UFLWJQRYJ8S6';
  var series_1_results = 4;
  var series_1_color = '#d62020';

  var series_2_channel_id = 1552376;  //Στοιχεία καναλιού μέσης υγρασίας
  var series_2_field_number = 1;
  var series_2_read_api_key = 'ZTK7QUL41XD2O23M';
  var series_2_results = 4;
  var series_2_color = '#00aaff';

  var chart_title = 'MinTemperature - AVGHumidity';     //Τίτλος του διαγράμματος
  var y_axis_title = 'Values';  //Τίτλος στον άξονα y

  var my_offset = new Date().getTimezoneOffset();  //Διαβάζει την ώρα του χρήστη
  // chart variable
  var my_chart;

  // when the document is ready
  $(document).on('ready', function() {
    // add a blank chart
    addChart();
    addSeries(series_1_channel_id,                    series_1_field_number, series_1_read_api_key, series_1_results, series_1_color);  //Προσθέτει στην πρώτη σειρά τα δεδομένα του πρώτου καναλιού
    addSeries(series_2_channel_id,                    series_2_field_number, series_2_read_api_key, series_2_results, series_2_color);  //Προσθέτει στη δεύτερη σειρά τα δεδομένα του δεύτερου καναλιού
  });

  // add the base chart
  function addChart() {
    var localDate;

    var chartOptions = {  //Χαρακτηριστικά για το γράφημα
      chart: {
        renderTo: 'chart-container',
        defaultSeriesType: 'column',
```

```
        backgroundColor: '#ffffff',
        events: { }
      },
      title: { text: chart_title },
      plotOptions: {
        series: {
          marker: { radius: 3 },
          animation: true,
          step: false,
          borderWidth: 0,
          turboThreshold: 0
        }
      },
      tooltip: {
        formatter: function() {  //Υπολογίζει την τοπική ώρα
          var d = new Date(this.x + (my_offset*60000));
          var  n  =  (this.point.name  ===  undefined)  ?  ''  :  '<br>'  +
this.point.name;
          return this.series.name + ':<b>' + this.y + '</b>' + n + '<br>' +
d.toDateString() + '<br>' + d.toTimeString().replace(/\(.*\)/, "");
        }
      },
      xAxis: {
        type: 'datetime',  //Δεδομένα στον άξονα x
        title: { text: 'Date' }
      },
      yAxis: { title: { text: y_axis_title } },  //Δεδομένα στον άξονα y
      exporting: { enabled: false },
      legend: { enabled: false },
      credits: {
        text: 'ThingSpeak.com',
        href: 'https://thingspeak.com/',
        style: { color: '#D62020' }
      }
    };
    my_chart  =  new  Highcharts.Chart(chartOptions);    //Δημιουργία  του
γραφήματος
  }
  function addSeries(channel_id, field_number, api_key, results, color) {
    var field_name = 'field' + field_number;

    // get the data with a webservice call
    $.getJSON('https://api.thingspeak.com/channels/'    +    channel_id    +
'/fields/' + field_number + '.json?offset=0&round=2&results=' + results +
'&api_key=' + api_key, function(data) {

      // blank array for holding chart data
      var chart_data = [];

      $.each(data.feeds, function() {
        var point = new Highcharts.Point();
        var value = this[field_name];
        point.x = getChartDate(this.created_at);
        point.y = parseFloat(value);
        if (this.location) { point.name = this.location; }
        if (!isNaN(parseInt(value))) { chart_data.push(point); }
      });

      my_chart.addSeries({          data:          chart_data,          name:
data.channel[field_name], color: color });
    });
```

```
  }

  function getChartDate(d) {
    return Date.parse(d) - (my_offset * 60000);
  }

</script>
```

Παρακάτω παρουσιάζεται αναλυτικά ο κώδικας που χρησιμοποιήθηκε για την καταγραφή θερμοκρασίας - υγρασίας, καρδιακών παλμών και αποστολή αυτών στο ThingSpeak - IoT.

Κώδικας για την προσθήκη βιβλιοθηκών

```
#include <ESP8266WiFi.h>;    //Προσθήκη βιβλιοθηκών

#include <WiFiClient.h>;

#include <ThingSpeak.h>;

#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321

#include "DHT.h"
```

Αρχικοποίηση των παραμέτρων για θερμοκρασία - υγρασία

```
// DHT Sensor Αρχικοποίηση παραμέτρων για θερμοκρασία - υγρασία

int DHTPin = 14; //D5
DHT dht(DHTPin, DHTTYPE);
float Temp_out;
float Humidity;
```

Σύνδεση WiFi , δήλωση καναλιού και δήλωση μεταβλητής για μέτρηση καρδιακών παλμών

```
const char* ssid = "COSMOTE-682082"; //Your Network SSID Σύνδεση WiFi

const char* pass = "Thimios0897"; //Your Network Password

int val;      //Δήλωση μεταβλητής για καρδιακούς παλμούς

int LDRpin = A0;   //LDR Pin Connected at A0 Pin


WiFiClient client;   //Δήλωση καναλιού
unsigned long myChannelNumber = 1438572; //Your Channel Number (Without
Brackets)
```

```
const char * myWriteAPIKey = "8LB3II36DJLC002Y";   //Write API Key from
ThingSpeak channel
```

Εκτύπωση των παραμέτρων του WiFi στη σειριακή οθόνη του Arduino IDE

```
Serial.print("Status: "); Serial.println(WiFi.status());    // Εκτύπωση
παραμέτρων WiFi σε σειριακή οθόνη Arduino IDE
  Serial.print("IP: ");     Serial.println(WiFi.localIP());
  Serial.print("Subnet: "); Serial.println(WiFi.subnetMask());
  Serial.print("Gateway: "); Serial.println(WiFi.gatewayIP());
  Serial.print("SSID: "); Serial.println(WiFi.SSID());
  Serial.print("Signal: "); Serial.println(WiFi.RSSI());
```

Διαβάζει και τυπώνει στη σειριακή οθόνη τη θερμοκρασία και την υγρασία

```
Temp_out = dht.readTemperature(); // Τιμή θερμοκρασίας
  Humidity = dht.readHumidity(); // Τιμή υγρασίας
  Serial.print("humidity: ");
  Serial.print(Humidity);         // Τυπώνει τιμή υγρασίας
  Serial.print("  Temp out: ");
  Serial.print(Temp_out);         // Τυπώνει τιμή θερμοκρασίας
```

Διαβάζει τους καρδιακούς παλμούς και τυπώνει την τιμή τους στη σειριακή οθόνη

```
val = analogRead(LDRpin); //Διαβάζει καρδιακούς παλμούς και τους αποθηκεύει
στη μεταβλητή val
val=val/10;
Serial.print(" heart: ");
Serial.println(val); // Τυπώνει καρδιακούς παλμούς
Serial.print("  ");
```

Στέλνει τα δεδομένα της θερμοκρασίας - υγρασίας και των καρδιακών παλμών στο ThingSpeak

```
ThingSpeak.writeField(myChannelNumber, 1,val, myWriteAPIKey); //Στέλνει
καρδιακούς παλμούς σε ThingSpeak
delay(20000);
ThingSpeak.writeField(myChannelNumber, 2,Temp_out, myWriteAPIKey);
//Στέλνει θερμοκρασία σε ThingSpeak
delay(20000);
ThingSpeak.writeField(myChannelNumber, 3,Humidity, myWriteAPIKey);
//Στέλνει υγρασία σε ThingSpeak
```

4.2 Σενάριο μέτρησης των βημάτων του χρήστη

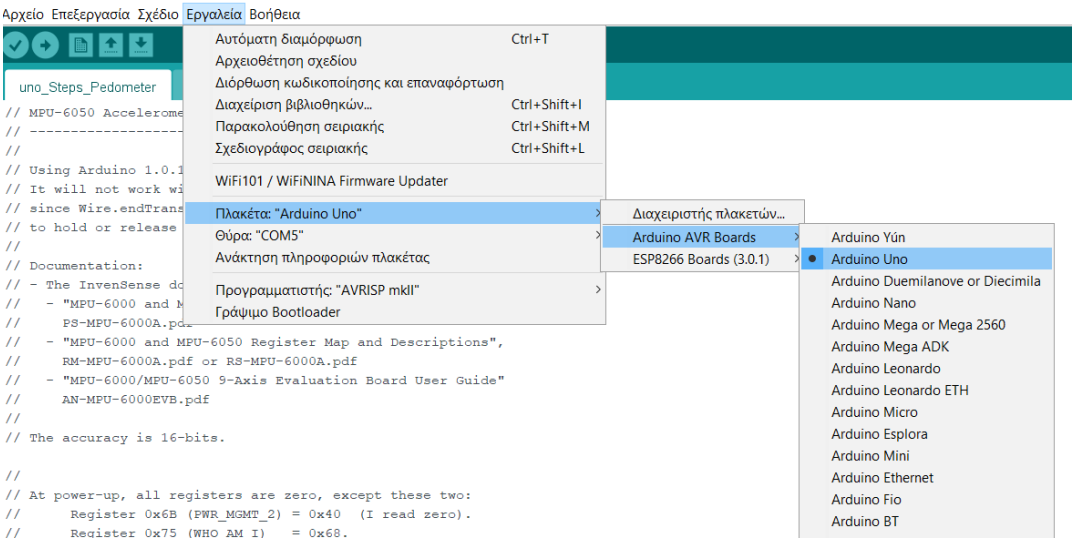
Το δεύτερο κύκλωμα περιλαμβάνει το Arduino Uno, το οποίο δέχεται μετρήσεις από τον αισθητήρα βημάτων και παρουσιάζεται στην παρακάτω εικόνα. Τα δεδομένα μπορούσαν να προβληθούν κι αυτά στη σελίδα ThingSpeak IoT αν υπήρχε διαθέσιμο NodeMCU που να έχει δύο αναλογικές θύρες και όχι μία όπως ισχύει στη συγκεκριμένη περίπτωση.

Τα υλικά που χρειάστηκαν στη συγκεκριμένη περίπτωση , είναι τα παρακάτω :

- Arduino Uno
- Επιταχυνσιόμετρο - γυροσκόπιο 3 αξόνων MPU 6050

Οι βιβλιοθήκες που χρειάστηκαν για αυτή την υλοποίηση είναι η Wire.h και η chris_pedometer.h .

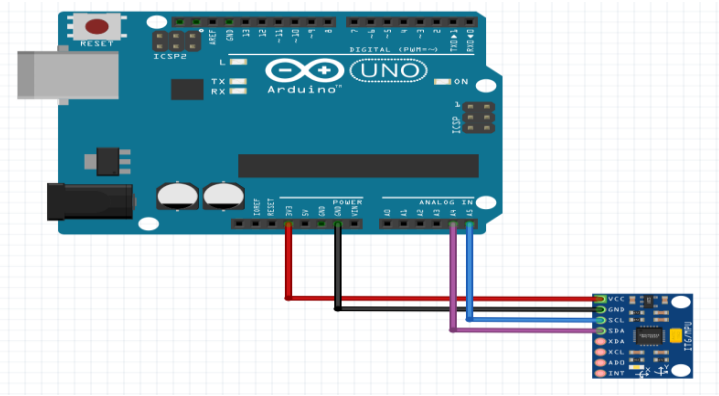
Όπως και στο προηγούμενο κύκλωμα ,έτσι και σε αυτό η επιλογή της πλακέτας γίνεται από τα παρακάτω βήματα.



Εικόνα 77: Επιλογή πλακέτας Arduino Uno

Πηγή : Πρόγραμμα Arduino IDE

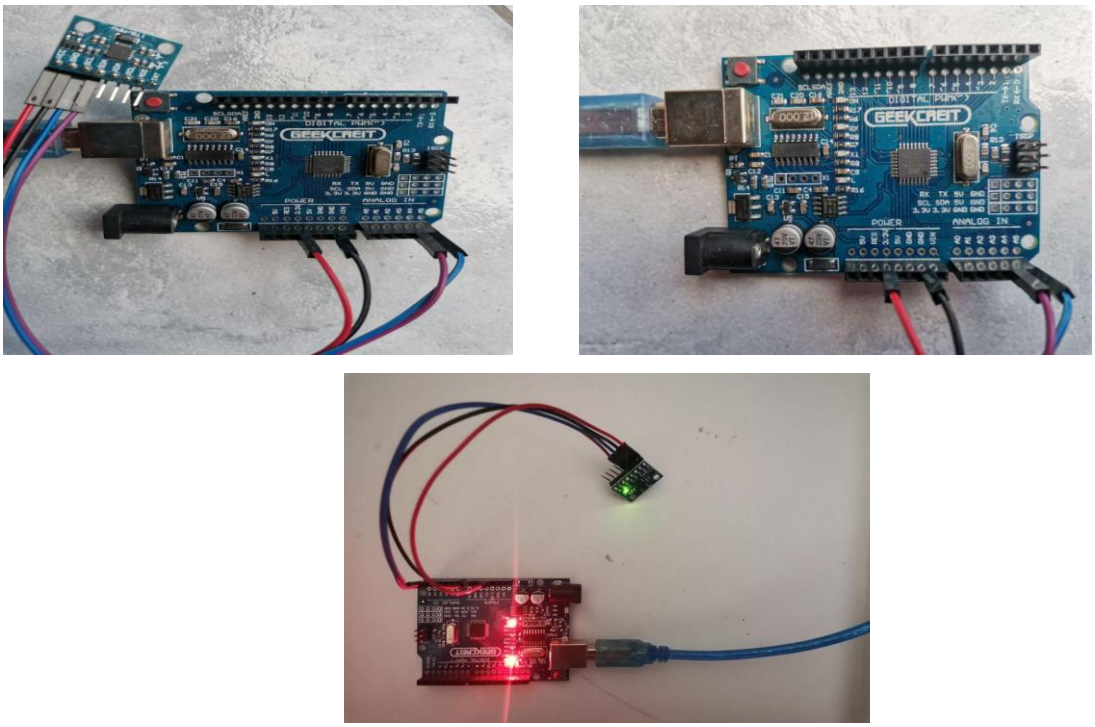
Παρακάτω απεικονίζεται το συγκεκριμένο κύκλωμα με τη χρήση του προγράμματος Fritzing.



Εικόνα 78: Κύκλωμα μέτρησης βημάτων με fritzing

Το κόκκινο καλώδιο VCC του αισθητήρα συνδέεται στο άκρο των 3 Volt της πλακέτας, το μαύρο καλώδιο Ground (GND) στο GND του Arduino UNO, το μπλε καλώδιο SCL στην αναλογική θύρα A5 και το μωβ καλώδιο SDA του MPu 6050 στην αναλογική θύρα A4 της πλακέτας.

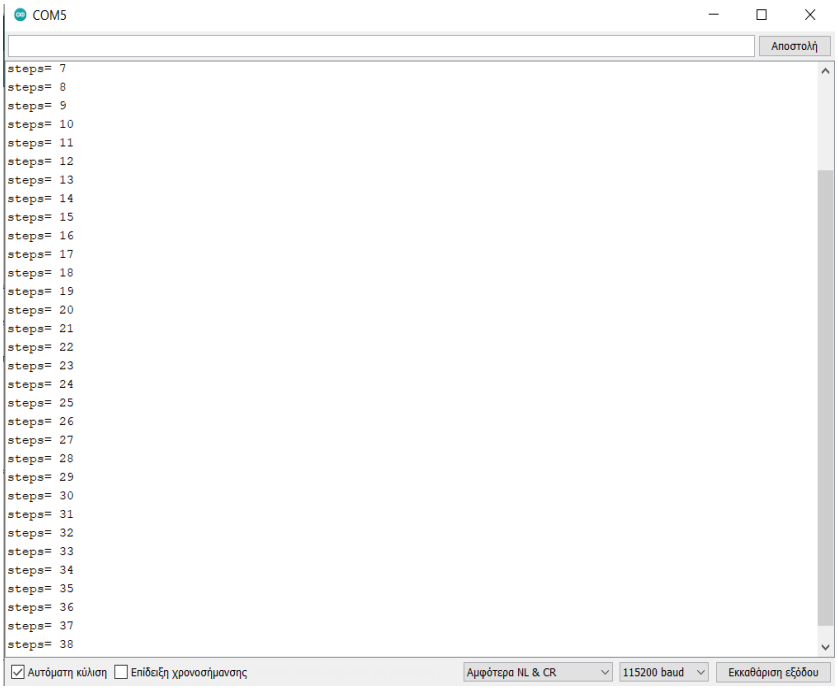
Παρακάτω παρουσιάζονται φωτογραφίες από το τελικό κύκλωμα.



Εικόνα 79: Κύκλωμα Arduino Uno, MPU-6050

Οι μετρήσεις που ακολουθούν αφορούν το δεύτερο κύκλωμα της εργασίας βάσει του οποίου γίνεται η μέτρηση των βημάτων. Τα αποτελέσματα της συγκεκριμένης υλοποίησης εμφανίζονται στη σειριακή οθόνη του Arduino IDE. Στο κάτω μέρος της οθόνης μπορεί να

επιλεγεί η επιθυμητή ταχύτητα (baud) από τη λίστα που εμφανίζεται, σύμφωνα με αυτή που θα επιλεχθεί και στον προγραμματισμό του Arduino.



Εικόνα 80: Μετρήσεις βημάτων

Ο κώδικας που χρησιμοποιήθηκε για την καταγραφή βημάτων του χρήστη παρουσιάζεται αναλυτικά παρακάτω.

Προσθήκη βιβλιοθηκών

```
#include <Wire.h>

#include "chris_pedometer.h"
```

Ορισμός τύπου παραμέτρων σε επιταχυνσιόμετρο, γυροσκόπιο και θερμοκρασία

```
typedef union accel_t_gyro_union
{
  struct
  {
    uint8_t x_accel_h;
    uint8_t x_accel_l;
    uint8_t y_accel_h;
    uint8_t y_accel_l;
    uint8_t z_accel_h;
    uint8_t z_accel_l;
```

```
    uint8_t t_h;
    uint8_t t_l;
    uint8_t x_gyro_h;
    uint8_t x_gyro_l;
    uint8_t y_gyro_h;
    uint8_t y_gyro_l;
    uint8_t z_gyro_h;
    uint8_t z_gyro_l;
  } reg;
```

Δήλωση μεταβλητών σε επιταχυνσιόμετρο και γυροσκόπιο

```
struct
  {
    int16_t x_accel;
    int16_t y_accel;
    int16_t z_accel;
    int16_t temperature;
    int16_t x_gyro;
    int16_t y_gyro;
    int16_t z_gyro;
  } value;
};
```

Ορισμός γωνίας περιστροφής του αισθητήρα

```
unsigned long last_read_time;
float         last_x_angle;  // Αυτές είναι οι φιλτραρισμένες γωνίες
float         last_y_angle;
float         last_z_angle;
float         last_gyro_x_angle;  // Αποθήκευση των γωνιών του γυροσκόπιου
για σύγκριση
float         last_gyro_y_angle;
float         last_gyro_z_angle;

inline unsigned long get_last_time() {return last_read_time;}
inline float get_last_x_angle() {return last_x_angle;}
inline float get_last_y_angle() {return last_y_angle;}
inline float get_last_z_angle() {return last_z_angle;}
inline float get_last_gyro_x_angle() {return last_gyro_x_angle;}
inline float get_last_gyro_y_angle() {return last_gyro_y_angle;}
inline float get_last_gyro_z_angle() {return last_gyro_z_angle;}
```

Βαθμονόμηση του επιταχυνσιόμετρου

```
float    base_x_accel;
float    base_y_accel;
float    base_z_accel;

float    base_x_gyro;
float    base_y_gyro;
float    base_z_gyro;

float x,y,z;
```

```
int count=0,prev=0;
int threshold=3;
```

Ο αισθητήρας διαβάζει τιμές

```
void loop()
{
  int error;
  double dT;
  accel_t_gyro_union accel_t_gyro;

  dT = ( (double) accel_t_gyro.value.temperature + 12412.0) / 340.0;

  // Διαβάζει τις αρχικές τιμές
  error = read_gyro_accel_vals((uint8_t*) &accel_t_gyro);

  // Διαβάζει το χρόνο για τους υπολογισμούς περιστροφής
  unsigned long t_now = millis();
```

Διαβάζει τιμές επιταχυνσιόμετρου

```
float accel_x = accel_t_gyro.value.x_accel;
  float accel_y = accel_t_gyro.value.y_accel;
  float accel_z = accel_t_gyro.value.z_accel;
```

Εμφάνιση των βημάτων στη σειριακή οθόνη

```
if(mag>=threshold && prev<threshold)
  {
        count+=1;
    Serial.print("steps= ");
    Serial.println(count);
  }

  prev = mag;
  x=angle_x;
  y=angle_y;
  z=angle_z;


  delay(100);
```

Βαθμονόμηση επιταχυνσιόμετρου και γυροσκόπιου

```
void calibrate_sensors() {
  int                   num_readings = 10;
  float                 x_accel = 0;
  float                 y_accel = 0;
  float                 z_accel = 0;
```

```
float                   x_gyro = 0;
float                   y_gyro = 0;
float                   z_gyro = 0;
accel_t_gyro_union      accel_t_gyro;
```

Κεφάλαιο 5ο

5.1 Συμπεράσματα και μελλοντικές επεκτάσεις

Χάρη στην παραπάνω πτυχιακή εργασία μου δόθηκε η ευκαιρία να γνωρίσω και να επεξεργαστώ την πλατφόρμα Arduino , η οποία σε συνδυασμό με την προσθήκη διαφόρων αισθητήρων έχει σαν αποτέλεσμα την κατασκευή ποικίλων και καινοτόμων εφαρμογών που αναφέρονται σε κλάδους ανάλογα με τη χρήση που επιθυμεί ο κατασκευαστής.

Το θέμα της πτυχιακής μου εργασίας ήταν η δημιουργία ενός “SmartWristband - Έξυπνου περικάρπιου” , το οποίο θα προσφέρει στο χρήστη ενημέρωση για τη θερμοκρασία - υγρασία του τόπου που είναι , τους καρδιακούς παλμούς του ατόμου καθώς και για τα βήματα που διανύει. Τέλος, οι πληροφορίες που αφορούν τη θερμοκρασία - υγρασία και τους καρδιακούς παλμούς προβάλλονται στη διαδικτυακή σελίδα ThingSpeak - IoT ενώ τα βήματα του χρήστη, λόγω έλλειψης υλικού-hardware, παρουσιάζονται στη σειριακή οθόνη του Arduino IDE.

Μελλοντικές επεκτάσεις:

Όσον αφορά τις μελλοντικές επεκτάσεις για την παραπάνω κατασκευή, χρήσιμο θα ήταν να προστεθούν αισθητήρες με τους οποίους η εφαρμογή θα αναβαθμιστεί και σαφέστατα θα εκτελεί περισσότερες λειτουργίες. Τέτοιοι αισθητήρες είναι ο αισθητήρας ποιότητας αέρα, ο αισθητήρας ανίχνευσης του μονοξειδίου του άνθρακα ή ο αισθητήρας αλκοόλ. Ακόμη, η εφαρμογή θα μπορούσε να ειδοποιεί το χρήστη με Email/SMS σε περίπτωση που οι καρδιακοί του παλμοί φτάσουν σε πολύ υψηλό επίπεδο.

ΠΑΡΑΡΤΗΜΑ Α

Κώδικας για την καταγραφή της θερμοκρασίας - υγρασίας, των καρδιακών παλμών και προβολή αυτών στο ThingSpeak - IoT.

```
#include <ESP8266WiFi.h>;    //Προσθήκη βιβλιοθηκών

#include <WiFiClient.h>;

#include <ThingSpeak.h>;


#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321


#include "DHT.h"

// DHT Sensor Αρχικοποίηση παραμέτρων για θερμοκρασία - υγρασία

int DHTPin = 14; //D5
DHT dht(DHTPin, DHTTYPE);
float Temp_out;
float Humidity;

const char* ssid = "COSMOTE-682082"; //Your Network SSID Σύνδεση WiFi

const char* pass = "Thimios0897"; //Your Network Password

int val;      //Δήλωση μεταβλητής για καρδιακούς παλμούς

int LDRpin = A0;   //LDR Pin Connected at A0 Pin


WiFiClient client;   //Δήλωση καναλιού


unsigned long myChannelNumber = 1438572; //Your Channel Number (Without
Brackets)


const char * myWriteAPIKey = "8LB3II36DJLC002Y";  //Write API Key from
ThingSpeak channel



void setup()

{

Serial.begin(115200);

//DHT
pinMode(DHTPin, INPUT);
```

```
dht.begin();

 WiFi.begin(ssid, pass);             // Σύνδεση σε WiFi router
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.print("Status: "); Serial.println(WiFi.status());    // Εκτύπωση
παραμέτρων WiFi σε σειριακή οθόνη Arduino IDE
  Serial.print("IP: ");     Serial.println(WiFi.localIP());
  Serial.print("Subnet: "); Serial.println(WiFi.subnetMask());
  Serial.print("Gateway: "); Serial.println(WiFi.gatewayIP());
  Serial.print("SSID: "); Serial.println(WiFi.SSID());
  Serial.print("Signal: "); Serial.println(WiFi.RSSI());

 ThingSpeak.begin(client);
  delay(2000);

}

void loop()

{
  Temp_out = dht.readTemperature(); // Τιμή θερμοκρασίας
  Humidity = dht.readHumidity(); // Τιμή υγρασίας
  Serial.print("humidity: ");
  Serial.print(Humidity);         // Τυπώνει τιμή υγρασίας
  Serial.print("  Temp out: ");
  Serial.print(Temp_out);         // Τυπώνει τιμή θερμοκρασίας

val = analogRead(LDRpin); //Διαβάζει καρδιακούς παλμούς και τους αποθηκεύει
στη μεταβλητή val
val=val/10;
Serial.print(" heart: ");
Serial.println(val); // Τυπώνει καρδιακούς παλμούς
Serial.print("  ");


ThingSpeak.writeField(myChannelNumber, 1,val, myWriteAPIKey); //Στέλνει
καρδιακούς παλμούς σε ThingSpeak
delay(20000);
ThingSpeak.writeField(myChannelNumber, 2,Temp_out, myWriteAPIKey);
//Στέλνει θερμοκρασία σε ThingSpeak
delay(20000);
ThingSpeak.writeField(myChannelNumber, 3,Humidity, myWriteAPIKey);
//Στέλνει υγρασία σε ThingSpeak

delay(20000);

}
```

ΠΑΡΑΡΤΗΜΑ Β

Κώδικας για την καταγραφή των βημάτων του χρήστη και προβολή αυτών στη σειριακή οθόνη

```
#include <Wire.h>

#include "chris_pedometer.h" //Προσθήκη βιβλιοθηκών

typedef union accel_t_gyro_union
{
  struct
  {
    uint8_t x_accel_h;
    uint8_t x_accel_l;
    uint8_t y_accel_h;
    uint8_t y_accel_l;
    uint8_t z_accel_h;
    uint8_t z_accel_l;
    uint8_t t_h;
    uint8_t t_l;
    uint8_t x_gyro_h;
    uint8_t x_gyro_l;
    uint8_t y_gyro_h;
    uint8_t y_gyro_l;
    uint8_t z_gyro_h;
    uint8_t z_gyro_l;
  } reg;
  struct
  {
    int16_t x_accel;
    int16_t y_accel;
    int16_t z_accel;
    int16_t temperature;
    int16_t x_gyro;
    int16_t y_gyro;
    int16_t z_gyro;
  } value;
};

unsigned long last_read_time;
float         last_x_angle;  // These are the filtered angles
float         last_y_angle;
float         last_z_angle;
float         last_gyro_x_angle;  // Store the gyro angles to compare drift
float         last_gyro_y_angle;
float         last_gyro_z_angle;


inline unsigned long get_last_time() {return last_read_time;}
inline float get_last_x_angle() {return last_x_angle;}
inline float get_last_y_angle() {return last_y_angle;}
inline float get_last_z_angle() {return last_z_angle;}
inline float get_last_gyro_x_angle() {return last_gyro_x_angle;}
inline float get_last_gyro_y_angle() {return last_gyro_y_angle;}
inline float get_last_gyro_z_angle() {return last_gyro_z_angle;}
```

```
float    base_x_accel;
float    base_y_accel;
float    base_z_accel;

float    base_x_gyro;
float    base_y_gyro;
float    base_z_gyro;

float x,y,z;
int count=0,prev=0;
int threshold=3;

void setup()
{
  int error;
  uint8_t c;

  Serial.begin(115200);


  Wire.begin();

error = MPU6050_read (MPU6050_WHO_AM_I, &c, 1);

error = MPU6050_read (MPU6050_PWR_MGMT_2, &c, 1);
MPU6050_write_reg (MPU6050_PWR_MGMT_1, 0);

calibrate_sensors();
  set_last_read_angle_data(millis(), 0, 0, 0, 0, 0, 0);
}


void loop()
{
  int error;
  double dT;
  accel_t_gyro_union accel_t_gyro;

  dT = ( (double) accel_t_gyro.value.temperature + 12412.0) / 340.0;

error = read_gyro_accel_vals((uint8_t*) &accel_t_gyro);

unsigned long t_now = millis();

float gyro_x = (accel_t_gyro.value.x_gyro - base_x_gyro)/FS_SEL;
  float gyro_y = (accel_t_gyro.value.y_gyro - base_y_gyro)/FS_SEL;
  float gyro_z = (accel_t_gyro.value.z_gyro - base_z_gyro)/FS_SEL;

float accel_x = accel_t_gyro.value.x_accel;
  float accel_y = accel_t_gyro.value.y_accel;
  float accel_z = accel_t_gyro.value.z_accel;

float RADIANS_TO_DEGREES = 180/3.14159;

float accel_angle_y = atan(-1*accel_x/sqrt(pow(accel_y,2) +
pow(accel_z,2)))*RADIANS_TO_DEGREES;
  float accel_angle_x = atan(accel_y/sqrt(pow(accel_x,2) +
pow(accel_z,2)))*RADIANS_TO_DEGREES;

float accel_angle_z = atan(sqrt(pow(accel_x,2) +
pow(accel_y,2))/accel_z)*RADIANS_TO_DEGREES;;
```

```
float dt =(t_now - get_last_time())/1000.0;
  float gyro_angle_x = gyro_x*dt + get_last_x_angle();
  float gyro_angle_y = gyro_y*dt + get_last_y_angle();
  float gyro_angle_z = gyro_z*dt + get_last_z_angle();

float unfiltered_gyro_angle_x = gyro_x*dt + get_last_gyro_x_angle();
  float unfiltered_gyro_angle_y = gyro_y*dt + get_last_gyro_y_angle();
  float unfiltered_gyro_angle_z = gyro_z*dt + get_last_gyro_z_angle();
float alpha = 0.96;
  float angle_x = alpha*gyro_angle_x + (1.0 - alpha)*accel_angle_x;
  float angle_y = alpha*gyro_angle_y + (1.0 - alpha)*accel_angle_y;
  float angle_z = gyro_angle_z;

set_last_read_angle_data(t_now, angle_x, angle_y, angle_z,
unfiltered_gyro_angle_x, unfiltered_gyro_angle_y, unfiltered_gyro_angle_z);

int mag=sqrt(pow(x-angle_x,2)+pow(y-angle_y,2)+pow(z-angle_z,2));

if(mag>=threshold && prev<threshold)
  {
        count+=1;
    Serial.print("steps= ");
    Serial.println(count);
  }

  prev = mag;
  x=angle_x;
  y=angle_y;
  z=angle_z;


  delay(100);

void calibrate_sensors() {
  int                   num_readings = 10;
  float                 x_accel = 0;
  float                 y_accel = 0;
  float                 z_accel = 0;
  float                 x_gyro = 0;
  float                 y_gyro = 0;
  float                 z_gyro = 0;
  accel_t_gyro_union    accel_t_gyro;

read_gyro_accel_vals((uint8_t *) &accel_t_gyro);


for (int i = 0; i < num_readings; i++) {
    read_gyro_accel_vals((uint8_t *) &accel_t_gyro);
    x_accel += accel_t_gyro.value.x_accel;
    y_accel += accel_t_gyro.value.y_accel;
    z_accel += accel_t_gyro.value.z_accel;
    x_gyro += accel_t_gyro.value.x_gyro;
    y_gyro += accel_t_gyro.value.y_gyro;
    z_gyro += accel_t_gyro.value.z_gyro;
    delay(100);
  }
  x_accel /= num_readings;
  y_accel /= num_readings;
  z_accel /= num_readings;
  x_gyro /= num_readings;
```

```
  y_gyro /= num_readings;
  z_gyro /= num_readings;

base_x_accel = x_accel;
  base_y_accel = y_accel;
  base_z_accel = z_accel;
  base_x_gyro = x_gyro;
  base_y_gyro = y_gyro;
  base_z_gyro = z_gyro;

int MPU6050_read(int start, uint8_t *buffer, int size)
{
  int i, n, error;

  Wire.beginTransmission(MPU6050_I2C_ADDRESS);
  n = Wire.write(start);
  if (n != 1)
    return (-10);

  n = Wire.endTransmission(false);    // hold the I2C-bus
  if (n != 0)
    return (n);


  Wire.requestFrom(MPU6050_I2C_ADDRESS, size, true);
  i = 0;
  while(Wire.available() && i<size)
  {
    buffer[i++]=Wire.read();
  }
  if ( i != size)
    return (-11);

  return (0);


int MPU6050_write(int start, const uint8_t *pData, int size)
{
  int n, error;

  Wire.beginTransmission(MPU6050_I2C_ADDRESS);
  n = Wire.write(start);
  if (n != 1)
    return (-20);

  n = Wire.write(pData, size);
  if (n != size)
    return (-21);

  error = Wire.endTransmission(true);
  if (error != 0)
    return (error);

  return (0);

int MPU6050_write_reg(int reg, uint8_t data)
{
  int error;

  error = MPU6050_write(reg, &data, 1);
```

```
  return (error);
}

void set_last_read_angle_data(unsigned long time, float x, float y, float
z, float x_gyro, float y_gyro, float z_gyro) {
  last_read_time = time;
  last_x_angle = x;
  last_y_angle = y;
  last_z_angle = z;
  last_gyro_x_angle = x_gyro;
  last_gyro_y_angle = y_gyro;
  last_gyro_z_angle = z_gyro;
}

int read_gyro_accel_vals(uint8_t* accel_t_gyro_ptr) {

accel_t_gyro_union* accel_t_gyro = (accel_t_gyro_union *) accel_t_gyro_ptr;

  int error = MPU6050_read (MPU6050_ACCEL_XOUT_H, (uint8_t *) accel_t_gyro,
sizeof(*accel_t_gyro));

uint8_t swap;
  #define SWAP(x,y) swap = x; x = y; y = swap

  SWAP ((*accel_t_gyro).reg.x_accel_h, (*accel_t_gyro).reg.x_accel_l);
  SWAP ((*accel_t_gyro).reg.y_accel_h, (*accel_t_gyro).reg.y_accel_l);
  SWAP ((*accel_t_gyro).reg.z_accel_h, (*accel_t_gyro).reg.z_accel_l);
  SWAP ((*accel_t_gyro).reg.t_h, (*accel_t_gyro).reg.t_l);
  SWAP ((*accel_t_gyro).reg.x_gyro_h, (*accel_t_gyro).reg.x_gyro_l);
  SWAP ((*accel_t_gyro).reg.y_gyro_h, (*accel_t_gyro).reg.y_gyro_l);
  SWAP ((*accel_t_gyro).reg.z_gyro_h, (*accel_t_gyro).reg.z_gyro_l);

  return error;
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. https://www.pharmacy2go.gr/el/%CF%80%CE%BF%CE%B9%CE%B1-%CE%B5%CE%B9%CE%BD%CE%B1%CE%B9-%CF%84%CE%B1-%CE%BF%CF%86%CE%B5%CE%BB%CE%B7-%CF%84%CE%B7%CF%82-%CE%B3%CF%85%CE%BC%CE%BD%CE%B1%CF%83%CF%84%CE%B9%CE%BA%CE%B7%CF%82-%CF%83%CF%84%CE%B7%CE%BD-%CF%85%CE%B3%CE%B5%CE%B9%CE%B1-%CE%BC%CE%B1%CF%82
2. https://www.iatronet.gr/askisi/gymnastiki-askisi/article/16337/gymnasteite-analoga-me-tin-ilikia-sas.html
3. https://www.iatropedia.gr/ygeia/fysiologikoi-sfygmoi-oi-times-ana-ilikia-kai-pos-na-tous-metrisete/49901/
4. https://www.onmed.gr/ygeia/story/365341/palmoi-kardias-oi-fysiologikes-times-ana-ilikia-pos-tha-toys-rixete
5. https://en.wikipedia.org/wiki/Fritzing
6. https://fritzing.org/
7. https://www.arduino.cc/en/software
8. https://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF_%CF%84%CF%89%CE%BD_%CF%80%CF%81%CE%B1%CE%B3%CE%BC%CE%AC%CF%84%CF%89%CE%BD
9. https://en.wikipedia.org/wiki/ThingSpeak
10. https://thingspeak.com/
11. https://thingspeak.com/channels/1438572/edit
12. https://en.wikipedia.org/wiki/NodeMCU
13. https://www.electroniclinic.com/nodemcu-esp8266-pinout-features-and-specifications/
14. https://elearn.ellak.gr/mod/book/view.php?id=2326
15. https://www.nodemcu.com/index_en.html
16. https://en.wikipedia.org/wiki/Notepad%2B%2B
17. https://notepad-plus-plus.org/downloads/
18. https://blogs.sch.gr/stekarapa/archives/12571
19. https://en.wikipedia.org/wiki/Pulse-width_modulation
20. https://el.wikipedia.org/wiki/%CE%93%CE%BB%CF%8E%CF%83%CF%83%CE%B1_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D_%CF%87%C

E%B1%CE%BC%CE%B7%CE%BB%CE%BF%CF%8D_%CE%B5%CF%80%CE%B9%CF%80%CE%AD%CE%B4%CE%BF%CF%85

21. https://el.wikipedia.org/wiki/%CE%93%CE%BB%CF%8E%CF%83%CF%83%CE%B1_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D_%CF%85%CF%88%CE%B7%CE%BB%CE%BF%CF%8D_%CE%B5%CF%80%CE%B9%CF%80%CE%AD%CE%B4%CE%BF%CF%85
22. https://el.m.wikipedia.org/wiki/Arduino
23. https://store.arduino.cc/products/arduino-uno-rev3
24. https://en.wikipedia.org/wiki/Arduino_Uno
25. https://en.wikipedia.org/wiki/NodeMCU
26. https://en.wikipedia.org/wiki/Single-board_microcontroller
27. https://www.electroniclinic.com/nodemcu-esp8266-pinout-features-and-specifications/
28. https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
29. https://store.arduino.cc/products/arduino-uno-rev3
30. https://store.arduino.cc/products/arduino-mega-2560-rev3
31. https://create.arduino.cc/projecthub/products/arduino-adk-rev-3
32. https://www.arduino.cc/en/Main/Arduino_BoardLeonardo
33. https://store-usa.arduino.cc/products/arduino-due
34. https://www.arduino.cc/en/Main.ArduinoBoardEsplora
35. https://www.arduino.cc/en/Main/ArduinoBoardEthernet/
36. https://www.arduino.cc/en/Main/ArduinoBoardYun/
37. https://www.arduino.cc/en/Main/ArduinoBoardBT?from=Main.ArduinoBoardBluetooth
38. https://www.google.com/search?q=arduino+pro+board&sxsrf=AOaemvKskLkligKdowjA3lICeGxCyZXt7Q%3A1635001758980&ei=niV0YYiuO4q-kwW2rIKoCw&ved=0ahUKEwiI9bOY6ODzAhUK36QKHTaWALUQ4dUDCA4&uact=5&oq=arduino+pro+board&gs_lcp=Cgdnd3Mtd2l6EAMyBAgAEBMyCAgAEBYQHhATMggIABAWEB4QEzIICAAQFhAeEBMyCAgAEBYQHhATMggIABAWEB4QEzIICAAQFhAeEBMyCAgAEBYQHhATMggIABAWEB4QEzIICAAQFhAeEBM6BwgjELADECc6BwgAEEcQsAM6BwgAELADEM6BggjECcQEzoFCAAQgAQ6BggAEBYQHjoECCMQJzoLCAAQgAQQsQMQgwFKBAhBGABQ5ghY_Cx

gsi9oAXACeAGAAckCiAHvGpIBCDAuMjEuMi4xmAEAoAEByAEKwAEB&sclient=gws-wiz
39. https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardFio
40. https://www.why.gr/%CE%BA%CE%B1%CF%84%CE%B1%CF%83%CF%84%CE%B7%CE%BC%CE%B1/open-hardware/arduino/arduino-main-boards/arduino-mini-05/?utm_source=Google%20Shopping&utm_campaign=google%20feed&utm_medium=cpc&utm_term=4558&gclid=CjwKCAjw5c6LBhBdEiwAP9ejG9dOQVIejb6xiqXn1NkcEXLNf9J-55E6ht9K643mg1DGesLN9YmiIxoCmXoQAvD_BwE
41. https://www.arduino.cc/en/Main/arduinoBoardDuemilanove
42. https://store.arduino.cc/products/arduino-micro
43. https://www.arduino.cc/en/guide/robot
44. https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoWiFiShield
45. https://store-usa.arduino.cc/products/arduino-motor-shield-rev3
46. https://learn.sparkfun.com/tutorials/gps-shield-hookup-guide/all
47. https://grobotronics.com/lcd-keypad-shield-for-arduino.html
48. https://projectmaniacs.wordpress.com/2014/11/22/%CE%B5%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE-%CF%83%CF%84%CE%BF-arduino/
49. https://el.wikipedia.org/wiki/%CE%9C%CE%B9%CE%BA%CF%81%CE%BF%CE%B5%CE%BB%CE%B5%CE%B3%CE%BA%CF%84%CE%AE%CF%82
50. https://el.wikipedia.org/wiki/%CE%91%CE%B9%CF%83%CE%B8%CE%B7%CF%84%CE%AE%CF%81%CE%B1%CF%82
51. https://grobotronics.com/rht03-dht22.html
52. https://www.sparkfun.com/products/11574
53. https://medium.com/@neurodatalab/every-beat-counts-comparing-remote-heart-rate-webcam-detector-to-wearables-d8d59aab863c
54. https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf